# GenesisBoard: ATSECTHI, Aether-Trained Self-Evolving Category Theory–HoTT Intelligence

Douglas C. Youvan

doug@youvan.com

August 11, 2025

ATSECTHI is a data-independent ("aether-trained") research program for growing intelligence from structural priors rather than corpora. The system inhabits a symbolic universe whose primitives are those of Category Theory and Homotopy Type Theory (HoTT): objects, morphisms, identities, associators, path spaces, and univalence-style probes. Instead of learning from labeled examples, ATSECTHI iteratively rewrites its own universe, optimizing a joint objective that rewards (i) compression via Minimum Description Length (MDL), (ii) satisfaction of categorical/HoTT coherence laws, and (iii) discovery of reproducible invariants ("motifs") that generalize across independently seeded runs. A self-predictor estimates the effect of candidate rewrites, a KL-budget controller maintains exploration, and optional 2-cells act as learned witnesses that stabilize composition. We propose falsifiable metrics—cross-seed invariant recurrence with MDL gain; phase-diagram stability under perturbation; interventional robustness—to distinguish genuine structure from noise or trivial fixed points. If successful, ATSECTHI establishes a third lane alongside statistical LLMs and theorem-provers: an intelligence that bootstraps lawful symbolic physics from minimal priors. If it fails to produce reproducible invariants with compression gains across seeds and curricula, the central claim is rejected.

Keywords: GenesisBoard, aether pretraining, self-evolving systems, Category Theory, Homotopy Type Theory, MDL, coherence loss, invariants, univalence probe, symbolic physics, emergence. 77 pages. A collaboration with GPT-5.

Note: Appendices contain executable Python code.

## I. Problem Statement and Positioning

### A. Why “aether training”: intelligence without corpora

Most contemporary systems inherit their behavior from external data. This bakes in human priors, linguistic shortcuts, and domain biases that obscure whether any *laws* would emerge without exposure to human artifacts. “Aether training” removes external corpora and instead seeds a typed symbolic universe with minimal structural priors—objects, morphisms, identities, composition, and lightweight HoTT probes. The system then evolves solely by internal rewrite dynamics that prefer lawful, compressible structure.

Goals:

- Isolate the contribution of **structure-seeking dynamics** from memorization.
- Test whether **local truths** emerge under categorical/HoTT constraints without semantic supervision.
- Produce **auditability**: every move is a typed rewrite with a costed effect on compression and coherence.

Operational stance:

- Start from randomness; allow only typed, legality-checked rewrites.
- Reward moves that jointly reduce description length and violation mass.
- Accept as “knowledge” only those invariants that recur across independently seeded universes.

### B. Limits of gradient-trained LLMs and goal-directed ATPs

**LLMs (statistical, corpus-trained).**
Strengths: broad coverage, impressive fluency, emergent skills. Limits: shortcut learning, latent contradiction, weak guarantees, brittle out-of-distribution reasoning, and no explicit ledger of *why* a representation was chosen.

Symptoms we avoid:

- **Data leakage of laws:** models appear to “discover” structure that was implicitly present in training corpora.
- **Unverifiable gains:** compression happens in hidden weights, not in an inspectable grammar.

**ATPs (theorem provers, goal-directed).**
Strengths: sound inference relative to a fixed axiom base; strong certificates. Limits: they do not *grow* their own worlds or priors; search is teleological (prove Φ) rather than ecological (improve universe fitness). Discovery is often hand-engineered (new lemmas, tactics) and depends on target theorems.

ATSECTHI's distinct bet:

- Replace **goal satisfaction** with **law pursuit**; the target is not a theorem but a universe whose internal contradictions shrink while its description shortens.
- Make **knowledge explicit** as reusable motifs and witnesses, not dispersed in parameters or tactic code.

**C. Hypothesis: coherence-seeking rewrite dynamics yield emergent intelligence**

**Hypothesis.** In a typed categorical/HoTT universe with minimal primitives, a rewrite process that jointly optimizes compression and coherence will produce reusable invariants (motifs) that transfer across seeds and curricula—constituting a primitive, domain-bounded intelligence.

Core signals (computed every step):

- Total objective: J = L_MDL + λ L_coh − ∑_{k} β_k Δ_k + γ R
    - L_MDL = codelength of the current universe's grammar and motif library.
    - L_coh = weighted sum of violations (associativity, identities, functoriality, naturality, higher-path consistency).
    - Δ_k = verified contribution of the k-th invariant to compression or error repair.
    - R = exploration/novelty regularizer under a KL budget.
- Violation mass: L_coh = α_assoc·E_assoc + α_id·E_id + α_fun·E_fun + …
- Compression jump per refactor: ΔL = L_MDL(before) − L_MDL(after)
- Cross-seed recurrence of a motif M: r = |M_seed ∩ M_ref| / |M_ref|
- Zero-data transfer gain on external categorical tasks: τ = score_with_M − score_without_M

Predictions:

1. **Phase structure.** As rewrite temperature decreases, universes move from noise → edge-of-chaos → ordered regimes with discrete compression jumps: ΔL ≫ 0 clustered in bursts.

2. **Invariant spectra.** Specific motifs (e.g., commutative triangles, diamond squares, stable 2-cell witnesses) recur with $r \geq r^*$ across seeds and shuffles.
3. **Transfer.** Recovered motifs lower $L_{MDL}$ and violation mass in *new* universes and improve zero-data performance on diagram-chasing micro-tasks: $\tau > 0$.

Falsifiers (any two suffice to reject):

- **No recurrence:** $r \approx 0$ for all candidate motifs across seeds.
- **No compression:** $E[\Delta L] \leq 0$ beyond small transient scales.
- **Brittle invariants:** motifs fail to reduce $L_{coh}$ or improve $\tau$ under perturbations or curricula changes.

Why this implies "emergent intelligence."
If lawful, compressive motifs arise reliably without external data and improve prediction/repair inside and across universes, the system exhibits the hallmarks of intelligence within its domain: self-generated abstraction, generalization via reuse, and counterfactual robustness—achieved by *seeking coherence*, not by imitating text or pursuing externally given theorems.

**You said:**

Expand: II. Minimal Priors ("The Aether") A. Syntax: objects, morphisms, identities, composition, functors B. HoTT layer: path types, higher paths, univalence-style probes (lightweight) C. Rewrite calculus: graph growth, contraction, and β-style normalization rules

**ChatGPT said:**

Thought for 34s

Great—here's a fuller Section II. All equations are in Word-Linear (UnicodeMath).

**II. Minimal Priors ("The Aether")**

**A. Syntax: objects, morphisms, identities, composition, functors**

**Universe signature.** The aether is a typed multigraph with a small algebraic signature:

- Sorts: Obj, Mor, optional Fun (functors between small categories).
- Typing maps: $s: Mor \to Obj$, $t: Mor \to Obj$.
- Identities: $id: Obj \to Mor$ with $s(id_A)=A \wedge t(id_A)=A$.
- Partial composition: $\circ: \{(g,f): t(f)=s(g)\} \to Mor$ with $s(g\circ f)=s(f) \wedge t(g\circ f)=t(g)$.

**Category laws (as soft constraints initially).**

- Associativity: (h ∘ g) ∘ f = h ∘ (g ∘ f).
- Identities: id_B ∘ f = f ∧ f ∘ id_A = f.

**Functors (optional from the start; unlockable via curriculum).**

- A functor F: $\mathcal{C} \to \mathcal{D}$ is (F_Obj, F_Mor) with:
    - Object map: F_Obj: Obj_$\mathcal{C}$ → Obj_$\mathcal{D}$.
    - Morphism map: F_Mor: Mor_$\mathcal{C}$ → Mor_$\mathcal{D}$ s.t.
      s(F_Mor(f)) = F_Obj(s(f)) ∧ t(F_Mor(f)) = F_Obj(t(f)).
- Preservation laws:
    - Identities: F_Mor(id_A) = id_{F_Obj(A)}.
    - Composition: F_Mor(g ∘ f) = F_Mor(g) ∘ F_Mor(f).

**Typed judgments (used by the legality checker).**

- Well-typed morphism: f: A→B ⇔ s(f)=A ∧ t(f)=B.
- Composability: g∘f defined ⇔ t(f)=s(g).

**MDL accounting units (names only; counts drive codelength).**
Obj, Mor, CompPairs, IdUse, FunSym, LawWitness, QuotEquiv, MotifRef
with an aether-side cost model like
L_MDL ≈ c_o|Obj| + c_m|Mor| + c_c|CompPairs| + c_id|IdUse| + c_f|FunSym| + c_w|LawWitness| − c_q|QuotEquiv| − c_r|MotifRef|.

**B. HoTT layer: path types, higher paths, univalence-style probes (lightweight)**

We enable only the fragments needed for detecting and reusing equivalences; this is **not** full HoTT.

**Identity/path types.**

- For a type (object space) A, the identity (path) type between elements a,a': A is Path_A(a,a').
  Constructors/operations:
    - Reflexivity: refl_a: Path_A(a,a).
    - Inverse: p^{-1}: Path_A(a',a) for p: Path_A(a,a').

- Concatenation: $p \cdot q$: Path_A(a,c) if p: Path_A(a,b) ∧ q: Path_A(b,c).

**Path laws (as rewrite equalities).**

- Left/right identity: $refl_a \cdot p = p \wedge p \cdot refl_b = p$.
- Inverse: $p \cdot p^{-1} = refl_a \wedge p^{-1} \cdot p = refl_b$.
- Associativity (paths): $(r \cdot q) \cdot p = r \cdot (q \cdot p)$.

**Lifting paths to categorical structure (witnesses).**

- When two parallel composites u,v: A→C are provably equal in normal form, attach a 2-cell witness ω: Path_{Hom(A,C)}(u,v).
- These witnesses are recorded in the invariant registrar and can later justify **contractions** (quotients) that reduce L_MDL.

**Univalence-style probe (lightweight heuristic).**

- Define a probe operator that proposes equivalences at the object level: probe(A,B) → cand_equiv(A,B) with a score σ.
- A candidate becomes an **equivalence** $A \simeq B$ only if it supports bidirectional transport that strictly reduces MDL without increasing violation mass:
  - Transport maps: tr_{A→B}, tr_{B→A} over attached hom-structure.
  - Acceptance test: $\Delta L_{MDL} > 0 \wedge \Delta L_{coh} \leq 0$.
  - If accepted, record $A \simeq B$ and enable quotienting rewrites guarded by witnesses.

**What we omit (by design).**

- No general Π/Σ types, no universes, no higher inductive types initially.
- No global univalence axiom; only localized, testable probes that act as **empirical** equivalences within the evolving universe.

**C. Rewrite calculus: graph growth, contraction, and β-style normalization rules**

ATSECTHI evolves by small, typed rewrites. Each rewrite R has preconditions, an effect on the universe, and a scored delta (ΔL_MDL, ΔL_coh).

**Growth rewrites (add structure).**

- add_obj: introduce a fresh A (costs +c_o).
- add_mor: add f: A→B with s(f)=A, t(f)=B (costs +c_m).

- compose: if t(f)=s(g), introduce h := g∘f: A→C (costs +c_c, may unlock later savings via normalization).
- add_id: ensure id_A exists and is linked (costs +c_id, may reduce coherence loss).
- add_functor: create F: $\mathcal{C}\to\mathcal{D}$ with minimal action on a subset of objects/morphisms (costs +c_f, enables preserved compositions).
- close_square: when u, v form opposite sides with shared ends, propose the missing edge to attempt a commuting square; attach tentative witness.

**Contraction rewrites (remove or merge structure).**

- cancel_id: f ∘ id_A → f, id_B ∘ f → f (saves c_id instances, reduces L_coh).
- assoc_nf: rewrite composites to a fixed bracketing (e.g., right-associated normal form) to canonicalize paths.
- merge_parallel: if witnesses certify u = v: A→C, delete one representative; redirect uses to the survivor (saves c_m and future c_c).
- quot_obj: if A ≃ B (accepted probe), merge objects and transport incident morphisms; attach global witness bundle (saves c_o and downstream costs).
- prune_dead: remove morphisms/objects with zero participation in any motif or composite (MDL saving with no coherence penalty).

**β-style normalization and η-style simplifications (term-level).**

- β-like: eliminate immediately reducible composites and identity detours to reach a **normal form** NF(·) for paths and morphism terms.
    - Example: NF((h ∘ g) ∘ f) = h ∘ (g ∘ f) (if right-assoc is chosen globally).
- η-like (extensionality-lite): if a functor F acts trivially on a subgraph (up to witnesses), replace F-image terms by originals to compress the grammar.

**Legality and scoring.**

- Legality predicate (typed + budgeted):
  legal(R,U) = 1 ⇔ R type-checks on U ∧ KL(p_t ‖ p_{t+1}) ≤ ε.
- Step delta:
  ΔJ = ΔL_MDL + λ ΔL_coh − ∑_{k} β_k Δ_k + γ ΔR.
- Commit rule: apply the top-scoring subset of candidate rewrites that remain mutually consistent; then run Normalize(·).

**Motif extraction (after normalization).**

- A **motif** is a minimal pattern whose presence predicts future MDL savings or coherence repair (e.g., a commuting diamond, a naturality square, a reusable 2-cell schema).
- Register motif M when its empirical utility passes a threshold:
  util(M) = E[ΔL_MDL | M] + ρ·E[−ΔL_coh | M] ≥ θ.
- Reuse via **MotifRef** symbols, which cut codelength whenever M reappears.

**Example micro-cycle (typed and local).**

1. add_mor f: A→B, add_mor g: B→C, compose → g∘f: A→C.
2. assoc_nf canonicalizes longer paths; cancel_id removes detours.
3. close_square proposes h: A→C when another route exists; ω: Path(h, g∘f) attached.
4. If ω recurs across seeds and reduces codelength via merge_parallel, elevate to a motif and enable automatic contraction on detection.

These priors are intentionally minimal: just enough syntax and lightweight higher structure to make **lawful compression** possible, measurable, and auditable—without importing semantics from any external corpus.

## III. Objective and Dynamics

### A. MDL term: L_MDL = L(universe) via grammar length

We encode the evolving universe as a compact "grammar" (symbols + rules + motif library + references). A practical two-part MDL splits into **model length** (the library and witnesses) and **data length** (the concrete graph/derivations encoded by that library).

- Decomposition:
  L_MDL = L_sym + L_rules + L_motiflib + L_witness + L_deriv − L_reuse
  - L_sym: objects/morphisms/functor symbols (universal codes for integers).
    Example: L_sym ≈ c_o |Obj| + c_m |Mor| + c_f |Fun|.
  - L_rules: production-like entries (e.g., permitted compositions).
  - L_motiflib: templates for recurring patterns (diamonds, naturality squares).
  - L_witness: stored 2-cells/paths that certify equalities.
  - L_deriv: residual cost to specify the current universe given the above.

    - L_reuse: savings from replacing repeated structures with MotifRef.

- Reuse accounting (per motif M):
  save(M) = usecount(M)·len(expansion(M)) − (len(M) + overhead)
  Aggregate into L_reuse = ∑_M save(M) (clipped at 0 if negative).

- Refactor gain for a step:
  ΔL_MDL = L_MDL(before) − L_MDL(after)

**Estimator choice.** Start with a **linear proxy** (the c_* costs above) for stability and speed; periodically validate with a heavier **enumerative code** (binomial/log terms) on snapshots to ensure the proxy tracks true codelength.

**B. Coherence loss: penalties for associativity/identity/functoriality violations**

L_coh aggregates typed constraint violations into a single scalar.

- Structure:
  L_coh = α_assoc·E_assoc + α_id·E_id + α_fun·E_fun + α_nat·E_nat + α_path·E_path

- Associativity (per composable triple (f,g,h)):
  Let u = NF((h ∘ g) ∘ f) and v = NF(h ∘ (g ∘ f)).
  Contribution: e = 0 if u = v; else e = 1 (or a distance if using term metrics).
  Then E_assoc = mean_triples(e)·w_triples (optionally weighted by usage).

- Identities: penalize missing identities and uncancelled detours.
  E_id = miss_id + detour_id, where detour_id counts residual id edges that normalization should have removed.

- Functoriality:
  For each F, check F(g ∘ f) vs F(g) ∘ F(f) and F(id_A) vs id_{F(A)}.
  E_fun = violations_fun (count or normalized rate).

- Naturality/path consistency (optional early, stronger later): mismatches in commuting squares or 2-cell path equalities.
  E_nat, E_path are counts or normalized rates after normalization.

Refactor effect per step:
ΔL_coh = L_coh(after) − L_coh(before) (note: improvements give negative ΔL_coh).

**C. Invariant reward: bonuses for motifs that compress and transfer across seeds**

Each discovered motif M_k accrues a **verified utility** that combines compression, coherence repair, and cross-seed transfer.

- Per-motif utility estimate:
  $\Delta_k = \mu_k^{MDL} + \rho \cdot \mu_k^{coh} + \eta \cdot \mu_k^{xfer}$
    - $\mu_k^{MDL} = E[\Delta L_{MDL} \mid apply(M_k)]$ (snapshot-averaged).
    - $\mu_k^{coh} = E[-\Delta L_{coh} \mid apply(M_k)]$ (positive if violations drop).
    - $\mu_k^{xfer} = g(r_k, \tau_k)$ where $r_k$ is recurrence and $\tau_k$ is zero-data transfer gain.
- Recurrence (cross-seed):
  $r_k = |M_k(seed) \cap M_k(ref)| \,/\, |M_k(ref)|$
- Zero-data transfer gain on external micro-tasks:
  $\tau_k = score_with(M_k) - score_without(M_k)$
- Trust-weighting (optional): scale by statistical confidence $conf_k$ so noisy motifs don't dominate.
  Replace $\Delta_k$ by $conf_k \cdot \Delta_k$, with $conf_k \in [0,1]$.

**Acceptance gate for promotion to the library.**
Promote $M_k$ only if $\Delta_k \geq \theta$ and $r_k \geq r^*$. Demote if utility decays below $\theta/2$.

**D. Total objective and step dynamics**

We **minimize** the scalar:
$J = L_{MDL} + \lambda L_{coh} - \sum_{k} \beta_k \Delta_k + \gamma R$

- Terms:
    - $L_{MDL}$: shorter is better.
    - $L_{coh}$: smaller violation mass is better ($\lambda \geq 0$).
    - $\Delta_k$: motif utilities; we subtract them so helpful motifs reduce J ($\beta_k \geq 0$).
    - R: exploration/regularization term (see below) with weight $\gamma$.

**Exploration term and budget.**
Use a KL-constrained proposal with an entropy bonus:

- Budget (hard or soft): $KL(p_t \,\|\, p_{t+1}) \leq \varepsilon$
- Bonus-penalty form: $R = H(p_{t+1}) - \tau \cdot KL(p_t \,\|\, p_{t+1})$
  This favors diverse proposals while limiting step-to-step drift.

**Per-step scoring.**
For a candidate rewrite set S (mutually consistent):

$\Delta J(S) = \Delta L_MDL(S) + \lambda\, \Delta L_coh(S) - \sum_{k \in lib} \beta_k\, \Delta_k(S) + \gamma\, \Delta R(S)$

Accept if either:

- Greedy: $\Delta J(S) < 0$, or
- Annealed: accept with probability $\exp(-\Delta J(S)/T)$ where T cools over time.

**Weight schedules (robust defaults).**

- Early **explore**: high γ, moderate ε, low λ to discover motifs even if messy.
- Mid **organize**: raise λ to enforce laws; lower γ and ε.
- Late **compress**: lower T, keep λ high; aggressive L_reuse via motif substitution.

**Legality/normalization pipeline per tick.**

1. Propose K typed rewrites from $p_{t+1}$ under the KL budget.
2. Predict ΔJ with the self-predictor; pick a consistent subset S.
3. Apply S, then run Normalize(·) (β-style/η-like simplifications).
4. Update metrics: ΔL_MDL, ΔL_coh, r_k, τ_k; refresh Δ_k.
5. Motif governance: promote/demote per thresholds; update L_reuse.
6. Log snapshot for audit (seed, RNG, costs, witnesses).

**Sanity checks and guards.**

- **Stagnation**: if $E[\Delta L_MDL] \approx 0$ and $E[\Delta L_coh] \approx 0$ over W windows, raise γ and temperature T, or inject counterexample generation.
- **Collapse**: if |Obj| or |Mor| drops below minima, temporarily suspend contractions; require fresh growth before further merges.
- **Overfitting to proxy**: periodically recompute a heavy MDL on snapshots; if drift is detected, recalibrate the c_* proxy.

**Worked micro-example (single tick).**

- Before: (A,B,C) with f: A→B, g: B→C, identity edges, no composite stored.
- Proposals S: {compose(g,f), cancel_id(B), assoc_nf}.

- Effects: adds h = g∘f, then normalizes; no violations introduced; one MotifRef(diamond) recognized later.
- Measured deltas: ΔL_MDL = +2 (add) − 3 (reuse after motif fires) = −1, ΔL_coh = −0.2 (fewer detours), Δ_k(diamond) = 0.4, ΔR ≈ 0.
- With λ = 1, β_diamond = 1, γ = 0:
  ΔJ = (−1) + 1·(−0.2) − 1·(0.4) = −1.6 → accept.

This objective/dynamics pairing keeps the search honest: **shorter codes**, **cleaner laws**, and **reusable motifs** must all move in the right direction (or be justified by exploration) for a step to stick.

## IV. Core Architecture

### A. Self-predictor: learned simulator of rewrite consequences

**Role.** Score candidate rewrites before committing by predicting ΔL_MDL, ΔL_coh, motif utilities Δ_k, and uncertainty.

**Inputs.**

- Local neighborhood around the rewrite (k-hop typed subgraph, normalized form IDs).
- Rewrite opcode + parameters (e.g., compose(g,f), merge_parallel(u,v)).
- Current cost proxy c_*, temperature T, budgets (ε, γ, λ, β_k).

**Outputs.**

- ΔJ_hat, ΔL_MDL_hat, ΔL_coh_hat, vector Δ_k_hat, uncertainty σ_hat.

**Model sketch.**

- Encoder: typed graph encoder (message passing with edge types {Mor, Id, Comp, 2Cell, Fun} and node roles {Obj, Term}).
- Head-1 (regression): predicts deltas.
- Head-2 (legality): legal_hat ∈ [0,1].
- Head-3 (uncertainty): variance via ensemble or dropout.

**Training (online DAgger-style).**

- Collect tuples (state, rewrite, true_deltas) from executed steps.

- Multitask loss (per sample):
  L_pred = |ΔL_MDL_hat − ΔL_MDL| + λ_coh·|ΔL_coh_hat − ΔL_coh| + κ·hinge_rank + ξ·BCE(legal_hat, legal_true)
  where hinge_rank = mean_max0(1 − (−ΔJ_hat_top − (−ΔJ_hat_other))).

**Use at runtime.**

- Rank K proposals by UCB = −ΔJ_hat − u·σ_hat, choose a consistent subset.

**B. KL-budget controller: exploration pressure without collapse**

We update the proposal policy over rewrites with a **trust region** so each step is close to the last but still exploratory.

**Constraint and update.**

- Budget: KL(p_t ‖ p_{t+1}) ≤ ε.
- Softmax reweighting: p_{t+1}(a) ∝ p_t(a) · exp(−ΔJ_hat(a) / τ) with line search on τ to satisfy the KL cap.
- Exploration term reported to the objective: R = H(p_{t+1}) − τ·KL(p_t ‖ p_{t+1}).

**Schedules.**

- Early explore: (ε, τ, γ) high.
- Mid organize: lower (ε, τ), raise λ.
- Late compress: small (ε, τ), anneal T.

**Guards.**

- If entropy H(p_{t+1}) drops below a floor, inject ε-greedy or add “counterexample” rewrites that create stress-tests.

**C. Invariant registrar: discovery, MDL accounting, and re-use**

**What is an invariant/motif.** A typed, canonicalized subgraph + witness schema whose presence predicts MDL savings or violation repair.

**Pipeline.**

1. **Detection.** After normalization, mine frequent typed patterns in recent windows; candidate M must occur ≥ s_min times.
2. **Canonicalization.** Use a deterministic labeling canon(M) (typed WL or equivalent) so motifs match across seeds.

3. **Utility estimation.**
   util(M) = E[ΔL_MDL | M] + ρ·E[−ΔL_coh | M] + η·μ^xfer(M)
   where μ^xfer(M) is measured on separate seeds/tasks.

4. **Promotion rule.** Promote if util(M) ≥ θ ∧ recurrence r(M) ≥ r*.
   Demote if utility decays (< θ/2) or harms transfer (negative μ^xfer).

5. **Library encoding.** Introduce MotifRef(M) with expansion map and witness hooks; update L_MDL via L_reuse.

6. **Portability audits.** On new seeds, measure r = |hits| / |expected| and zero-data gain τ; log to the ledger.

**Accounting hooks.**

- Per-motif fields: (len(M), overhead, usecount, ΔL_MDL_running, ΔL_coh_running, r, τ, conf).
- Savings for a commit set S:
  Δ_k(S) = conf_k · (μ_k^MDL + ρ·μ_k^coh + η·μ_k^xfer).

**D. Witness 2-cells: learned certificates that stabilize compositions**

**Purpose.** Turn equalities that recur in practice into **certified** contractions, reducing ambiguity and coherence loss.

**Kinds of witnesses.**

- **Path witnesses** ω: Path_{Hom(A,C)}(u,v) attaching to parallel arrows.
- **Square/diamond witnesses** certifying commuting diagrams (naturality-like).
- **Equivalence bundles** for A ≃ B (object-level merges with transport).

**Generation & verification.**

1. **Propose** (from detector or predictor) a witness schema for (u,v).
2. **Verify** by normalizing both sides; if NF(u)=NF(v), attach trivial refl.
   Else, accept a **learned witness** only if:
   ΔL_MDL > 0 ∧ ΔL_coh ≤ 0 ∧ no_new_violations.
3. **Confidence.** Each witness stores (support, conflicts, conf); only high-conf witnesses participate in automatic contractions.

4. **Use in rewriting.**
    - merge_parallel(u,v) allowed when a qualifying ω exists.
    - During functorial checks, witnesses can justify equalities on images.

**Decay and conflict handling.**

- If later steps produce conflicts (counterexamples), decrement conf; below a floor, quarantine or retire the witness and undo dependent merges via rollback snapshots.

**E. Snapshotter: time-stamped universes for phase analysis and replay**

**Why.** Reproducibility, phase-diagram science, and rollback for red-teaming.

**Granularity.**

- **Tick snapshots** (cheap): hashes of sets (Obj, Mor, Comp, Motifs, Witnesses), costs, metrics (L_MDL, L_coh, ΔJ, entropy, temperature).
- **Epoch snapshots** (heavier): full serializations of the universe, library, predictor weights (optional), and RNG state.

**Schema (JSONL, one line per snapshot).**

json

CopyEdit


{

"run_id": "ATSECTHI-20250811-063000",

"seed": 12345,

"tick": 51215,

"timestamp": "2025-08-11T06:30:00-05:00",

"costs": {"L_MDL": 102345.7, "L_coh": 217.4, "J": 102563.9},

"params": {"λ":1.0,"γ":0.2,"ε":0.05,"T":0.12},

"entropy": 2.31,

"events": ["compose(g,f)","assoc_nf","merge_parallel(h, g∘f)"],

"motifs": [{"id":"M_diamond","usecount":812,"Δ":0.41,"r":0.73}],

"witnesses": [{"lhs":"u","rhs":"v","conf":0.92}],

"hash": "u:abcd...; m:ef01...; lib:9876..."

}

**Phase analysis metrics (per window).**

- Compression jump: ΔL = L_MDL(before) − L_MDL(after).
- Violation mass: L_coh.
- Motif density: |active_motifs| / |Mor|.
- Policy entropy: H(p_t).
- Edge-of-chaos indicator: variance of ΔJ and burstiness of ΔL (e.g., Fano factor).

**Replay & rollback.**

- Deterministic re-execution from (seed, manifest of rewrites) reproduces states bit-for-bit.
- **Branching replays**: apply alternative subsets at a saved tick to test counterfactuals.
- **Safety**: signed hashes ensure library/witness integrity; any mismatch halts replay.

---

**Minimal end-to-end tick (inside one process)**

1. **Propose** K typed rewrites from p_t.
2. **Predict** ΔJ_hat with self-predictor; compute UCB.
3. **Select** consistent subset S under KL-budget update.
4. **Apply** S, then **Normalize** (β/η).
5. **Measure** true deltas; **Train** predictor on fresh tuples.
6. **Detect** motifs; **Promote/Demote** in registrar; update L_reuse.
7. **Attach/Update** witnesses and resolve safe contractions.
8. **Snapshot** tick; every M ticks, take an epoch snapshot.

This gives you a concrete blueprint to implement incrementally without external data: start with the snapshotter + legality + simple predictor (linear proxy), then layer in registrar, witnesses, and KL-control.

## V. Training Regimen (No External Data)

### A. Bootstrap: random small universes with identity and partial composition

**Seed distribution (per run).**

- Objects: $|\mathrm{Obj}| \sim \mathrm{Uniform}\{3,\ldots,8\}$.
- Primitive morphisms: sample edges $f: A \to B$ with $p_{edge} \in [0.2,0.4]$, avoid self-loops initially.
- Identities: ensure $id_A$ for all $A$.
- Partial composition index: create CompPairs = { (g,f): t(f)=s(g) } but **do not** materialize $g \circ f$ yet; let rewrites add them.

**Initial costs/weights (stable defaults).**

- $c_o=6$, $c_m=8$, $c_c=4$, $c_{id}=2$, $c_f=16$.
- $\lambda=0.3$, $\gamma=0.5$, $\varepsilon=0.10$, $T=0.25$.

**Normalization.** Choose a global bracketing (e.g., right-associated) and a canonical name scheme so NF(·) is unique.

### B. Self-play rewriting: propose/evaluate/apply under J

**Tick loop (per run).**

1. **Propose** K rewrites from policy $p_t$ (typed): {add_obj, add_mor, add_id, compose, cancel_id, assoc_nf, close_square, merge_parallel, quot_obj, prune_dead}.
2. **Score** each with self-predictor → $\Delta J_{hat}$, $\sigma_{hat}$.
3. **Select** a consistent subset S via UCB = $-\Delta J_{hat} - u \cdot \sigma_{hat}$.
4. **Enforce KL**: update $p_{t+1}$ s.t. $KL(p_t \| p_{t+1}) \le \varepsilon$.
5. **Apply + Normalize**, then **measure** true $\Delta L_{MDL}$, $\Delta L_{coh}$, $\Delta_k$.
6. **Train** predictor on tuples; **log** snapshot.

**Annealing (suggested schedules).**

- Temperature: $T_t = T_0 \cdot \exp(-t / \tau_T)$ with $\tau_T \approx 1e5$ ticks.
- KL budget: $\varepsilon_t = \varepsilon_0 \cdot \exp(-t / \tau_\varepsilon)$; start loose, tighten.
- Exploration weight: reduce $\gamma$ as $H(p_t)$ stabilizes.

**C. Local-truth curriculum: progressively unlock laws**

Stage gates are *data-free*, triggered by internal metrics. Let window-average be $\langle\cdot\rangle_W$.

**Stage 0 – Compose & Normalize (discover structure).**

- Low λ; allow growth/merges; mine early motifs.
- Gate to Stage 1 when $\langle E_{id}\rangle_W \le \varepsilon_{id}^*$ and $|Mor|/|Obj| \ge r_{min}$.

**Stage 1 – Enforce Identities.**

- Raise $\alpha_{id}$; push detours to zero.
- Gate to Stage 2 when $\langle E_{id}\rangle_W \approx 0$ and $\Delta L_{MDL}$ still shows bursts.

**Stage 2 – Associativity Discipline.**

- Raise $\alpha_{assoc}$; ensure NF-consistency.
- Gate to Stage 3 when $\langle E_{assoc}\rangle_W \le \varepsilon_{assoc}^*$.

**Stage 3 – Functoriality (optional unlock).**

- Enable add_functor with initial tiny action sets.
- Penalize violations: E_fun.
- Gate to Stage 4 when $\langle E_{fun}\rangle_W \le \varepsilon_{fun}^*$ and motifs show portability $\langle r\rangle_W \ge r^*$.

**Stage 4 – Naturality & 2-Cell Witnesses.**

- Turn on square/diamond checks and path witnesses.
- Promote contractions guarded by high-confidence witnesses.

**Typical thresholds.**

- $\varepsilon_{id}^* = 0$, $\varepsilon_{assoc}^* \in [0, 1e{-}3]$ (rate), $\varepsilon_{fun}^* \in [0, 1e{-}3]$, $r^* \in [0.6,0.8]$.

**D. Counterexample-driven repair: adversarial generation of violations**

Inject *hard cases* that locally raise L_coh to stress-test repair.

**Generators (called intermittently).**

- **Assoc-breaker:** pick composable (f,g,h); create rival composite forms and withhold witnesses so E_assoc spikes; reward rewrites that restore equality in NF.
- **Id-detourer:** insert identity detours in high-use paths; system must cancel_id without harming MDL.

- **Functor-breaker:** for active F, introduce x = F(g ∘ f) and y = F(g) ∘ F(f) with slight asymmetry; push creation of the missing edge/witness to commute.
- **Square-glitcher:** build near-commuting diamonds with one noisy edge; measure if witnesses emerge and merge_parallel fires.

**Inner objective for generator G:** maximize difficulty under budgets:

- max_G λ L_coh − κ ΔL_MDL s.t. KL(p_gen ‖ base) ≤ ε_gen.
- Accept adversarial insertions only if they don't collapse universe size bounds.

**Repair metrics (windowed).**

- Time-to-fix: τ_fix = argmin Δt : E_violation drops by 90%.
- Overfix penalty: ensure repair did not inflate L_MDL beyond δ_MDL*.

**E. Cross-universe transfer: measure motif portability across seeds**

Hold out seeds and curricula variants to test reuse.

**Protocol.**

1. **Discover** motif library Lib_train on seed set S_train.
2. **Freeze** Lib_train; start fresh seeds S_test with empty graphs but *motif references enabled*.
3. **AB test** on S_test: with vs without Lib_train.

**Metrics.**

- Recurrence: r = |M_seed ∩ M_ref| / |M_ref|.
- Transfer gain on internal metrics:
  ΔMDL_xfer = L_MDL(noLib) − L_MDL(withLib) (same tick budget).
  Δcoh_xfer = L_coh(noLib) − L_coh(withLib).
- External micro-task gain (diagram chasing, rewrite puzzles):
  τ = score_withLib − score_noLib.

**Survival & drift.**

- Track motif survival over stages: surv_t(M) = 1 if active; else 0.
- Retire motifs whose util(M) decays (< θ/2) or that show **negative transfer** (τ < 0).

**Portability audit schedule.**

- Every R ticks, sample a motif batch, run quick-fit on 3 held-out seeds (short budgets), log (r, ΔMDL_xfer, τ); adjust β_k weights by confidence.

---

**Suggested run plan (toy→mid).**

- Toy: 32 seeds × 2e5 ticks, Stages 0–2 only; validate ΔL bursts and r.
- Mid: 16 seeds × 5e5 ticks, unlock Stage 3–4; enable adversarial generators every 5e3 ticks; perform AB transfer every 2e4 ticks.

This regimen keeps the system strictly data-free while forcing it to *earn* structure: first by finding compressive patterns, then by passing graded law-checks, repairing adversarial breaks, and finally proving motifs portable across fresh universes.

## VI. Emergence Measurements

### A. Motif statistics: diamonds, commutative triangles, higher homotopies

**Canonical detection (after Normalize).**

- Commutative triangle A →^f B →^g C with h: A→C and witness ω: Path(h, g∘f).
- Diamond (square) with parallel routes u, v: A→D and ω: Path(u, v).
- Higher homotopy: chains of 2-cells ω_1: Path(u, v), ω_2: Path(ω_1, ω_1′), etc.

**Windowed counts (window W ticks).**

- Triangle count: C_△ = |{ comm_triangles in W }|
- Diamond count: C_◇ = |{ diamonds in W }|
- Higher path count (level ≥ 2): C_2+ = |{ 2-cell chains length ≥ 2 in W }|

**Densities (size-normalized).**

- Triangle density: ρ_△ = C_△ / |Mor|
- Diamond density: ρ_◇ = C_◇ / |Mor|
- Higher-path density: ρ_2+ = C_2+ / |Mor|

**Stability/portability.**

- Cross-seed recurrence of motif type M: r_M = |M_seed ∩ M_ref| / |M_ref|
- Promotion utility (running): util(M) = E[ΔL_MDL | M] + ρ·E[−ΔL_coh | M]

**Null-model baselines.** Recompute ρ_△, ρ_◇, ρ_2+ on (i) label-shuffled graphs and (ii) degree-preserving random rewires; report z_M = (ρ_M − ρ_M^null) / σ_M^null.

**B. Phase diagrams: regimes of order, edge-of-chaos, noise**

**Control parameters (axes).** Temperature T, KL budget ε, coherence weight λ (others fixed). Sweep a grid and average over seeds.

**Order parameters.**

- Compressibility per edge: κ = L_MDL / |Mor|
- Violation mass: ϵ = L_coh / |CompPairs|
- Motif activity: μ = |active_motifs| / |Mor|
- Policy entropy: H = −∑_a p(a) log p(a)

**Edge-of-chaos indicators (burstiness & sensitivity).**

- MDL jump Fano factor in window W: F = Var(ΔL) / E[ΔL] (on ΔL>0 events)
- ΔJ volatility: σ_J = Std(ΔJ)
- Local sensitivity: perturb a single accepted rewrite; measure
  s = |J(U′) − J(U)| after k normalization steps
- EOC index (unitless composite, rescaled to [0,1]):
  EOC = ζ_1·norm(F) + ζ_2·norm(σ_J) + ζ_3·norm(s)

**Regime labeling (heuristic).**

- **Noise:** μ ≈ 0 ∧ ϵ high ∧ κ high ∧ H high
- **Edge-of-chaos:** μ rising ∧ F mid–high ∧ ϵ decreasing ∧ κ decreasing
- **Order:** μ plateau ∧ ϵ low ∧ κ low ∧ H low–mid

**Diagram construction.** For each (T, ε, λ), record (κ, ϵ, μ, EOC); plot iso-contours to locate the EOC ridge where reusable motifs are maximal but violations remain controlled.

**C. Compression jumps: discrete MDL drops indicating re-factoring events**

**Per-tick MDL delta.**

- $\Delta L_t = L_{MDL}(t-1) - L_{MDL}(t)$ (positive means compression)
- Event threshold: $\Delta L_t \geq \theta_L$ (e.g., $\theta_L = 3$ proxy bits)

**Burst segmentation.**

- A **burst** is a maximal consecutive run with $\Delta L_t \geq \theta_L$.
- Burst size: $S = \sum_{t \in burst} \Delta L_t$
- Inter-burst interval: IBI = gap length between bursts

**Associations.**

- Correlate bursts with motif events and witness promotions:
  corr(S, count_promotions_in_burst);
  corr($\Delta L_t$, $\sum_k \Delta_k(t)$)
- Change-point tests (CUSUM on $L_{MDL}$): flag structural refactors aligned to library updates.

**Scaling and tails.**

- Fit $P(S \geq s) \sim s^{-\alpha}$ on log-log; edge-of-chaos often shows heavy tails ($\alpha \in [1.5, 3]$). Report $\alpha$ with goodness-of-fit and null comparisons (shuffled timelines).

**D. Predictive validity: self-predictor accuracy vs. universe complexity**

**Targets.** $\Delta J$, $\Delta L_{MDL}$, $\Delta L_{coh}$, top-k ranking.

**Error metrics.**

- MAE: $MAE(\Delta J) = \langle |\Delta J_{hat} - \Delta J| \rangle$
- Rank correlation: $\rho_{rank} = Spearman(-\Delta J_{hat}, -\Delta J)$
- Top-k precision: $Prec@k = |Top_k_{hat} \cap Top_k_{true}| / k$
- Calibration (binning by $\Delta J_{hat}$): expected vs. observed $\Delta J$

**Complexity index (composite).**

- $\chi = a_1 \log(1 + |Obj|) + a_2 \log(1 + |Mor|) + a_3 \mu + a_4 \langle path_len \rangle + a_5 |Lib|$
- Bucket by $\chi$ quartiles; report each metric vs. $\chi$.

**Scaling laws (fit on snapshots).**

- MAE(ΔJ | χ) ≈ c_0 + c_1 · χ^α
- ρ_rank(χ) ≈ r_∞ − b · χ^β
- Acceptable regime: MAE sublinear or slowly rising (α ≤ 0.5), ρ_rank ≥ 0.6 through Q3.

**Ablations for trust.**

- Freeze predictor (no learning) vs. online updates → Δ performance
- UCB vs. greedy selection → effect on Prec@k and ΔL_MDL per tick
- Uncertainty off (u=0) vs. tuned → overexploration/underexploration diagnostics

**Decision usefulness.**

- Net gain from prediction:
  Δ_gain = E[ΔJ_with_predictor] − E[ΔJ_random] (same KL and temperature)
- If Δ_gain > 0 across χ buckets and Prec@k remains high, the predictor is validated as *actionable*, not merely accurate.

---

**Reporting template (per run set).**

- Motifs: (ρ_△, ρ_◇, ρ_2+, z-scores, r_M)
- Phases: heatmaps of (κ, ϵ, μ, EOC) over (T, ε) at fixed λ
- Compression: burst distribution, α, CUSUM points, motif-link correlations
- Prediction: MAE(ΔJ), ρ_rank, Prec@k vs. χ, Δ_gain

These measurements together tell a coherent story: *what* structures are appearing (motifs), *where* in parameter space emergence is strongest (phase diagrams), *how* the codebase is reorganizing itself (compression bursts), and *whether* the internal simulator is good enough to steer growth as universes become more complex (predictive validity).

## VII. Risks & Falsification

### A. Risk: trivial fixed points or explosive noise

**Failure modes.**

- **Trivial fixed points (collapse).** The universe stops changing except for superficial rewrites; motifs stall; policy entropy collapses.
  Indicators (window W):
  $\langle \Delta L_MDL \rangle_W \approx 0 \wedge \langle \Delta L_coh \rangle_W \approx 0 \wedge \mu \approx 0 \wedge H(p_\{t\}) \leq H_min$
- **Explosive noise (divergence).** Unbounded growth with little compression; violations persist; motifs do not stabilize.
  Indicators:
  $\kappa = L_MDL / |Mor|$ rises or plateaus high, $\epsilon = L_coh / |CompPairs|$ high, $\mu \approx 0$, $H(p_t)$ near max, and $F = Var(\Delta L) / E[\Delta L] \gg 1$ with tiny $E[\Delta L]$.

**Guards.**

- Anti-collapse: temporarily raise $\gamma$, $\varepsilon$, $T$, inject adversarial counterexamples, or widen candidate operator mix.
- Anti-noise: raise $\lambda$, lower $\gamma$, $\varepsilon$, $T$, and enable stricter legality filters.

If a run spends $\geq q\%$ of its epochs in either zone despite guard schedules, label the run **non-emergent**.

### B. Falsification: absence of recurrent invariants with MDL gains across seeds/shuffles

**Core claim under test.** Motifs recur across independently seeded universes and yield positive compression and coherence repair.

**Pre-registered acceptance thresholds (example).**

- Seeds: $N \geq 64$, shuffled rewrite orders per seed: $K \geq 3$.
- For a candidate motif type $M$:
  - **Recurrence:** $r_M = |M_seed \cap M_ref| / |M_ref| \geq r^*$ in $\geq 60\%$ of seeds, with $r^* \in [0.6, 0.8]$.
  - **Compression:** $\mu_M^{MDL} = E[\Delta L_MDL \mid apply(M)] \geq \delta_MDL^*$, e.g., $\delta_MDL^* = 1.0$ proxy bits per use.
  - **Coherence repair:** $\mu_M^{coh} = E[-\Delta L_coh \mid apply(M)] \geq 0$.

- Hypothesis tests: for each M, test H0: $r_M < r^* \vee \mu_M^{MDL} \le 0$. Reject H0 at $\alpha = 0.01$ (Benjamini–Hochberg FDR across tested motifs).

**Null baselines.**

- **Label shuffles** and **degree-preserving rewires**; compute $z_M = (\rho_M - \rho_M^{null}) / \sigma_M^{null}$. Require $z_M \ge 3$ for promotion.

**Falsification rule.** If no motif class satisfies the thresholds and tests above across seeds and shuffles, the emergence claim is **rejected** for that configuration (and, if robust to hyperparameters, rejected overall).

**C. Stress tests: curriculum ablations, noisy rewrites, law permutations**

**Purpose.** Ensure motifs are law-driven, not artifacts of curriculum order, operator mix, or proxy cost.

**Ablations (one factor at a time).**

- **No-MDL:** set motif reuse off ($L_{reuse} = 0$), measure drop in motif survival and density $\rho_M$.
- **No-coherence:** set $\lambda = 0$, verify motifs degrade to frequency without repair effect ($\mu_M^{coh} \to 0$).
- **No-witness:** disable 2-cells; merging must cease; if "motifs" still appear with similar $\Delta L_{MDL}$, they are likely spurious.
- **No-KL:** remove trust region; expect explosion or collapse; confirm dependence on exploration control.

**Law permutations / curriculum swaps.**

- Swap stage order: Associativity→Identity vs Identity→Associativity; introduce Functoriality early vs late.
  Criterion: genuine motifs persist (within tolerance) regardless of stage order; if motif spectra shift radically, they may be curriculum artifacts.

**Operator mix perturbations.**

- Vary proposal priors over {add, compose, cancel_id, merge_parallel, quot_obj,…} by ±50%.
  Criterion: motif densities should vary smoothly; brittle dependence indicates overfitting to operator frequencies.

**Noisy rewrites (stress).**

- Accept a fraction $\nu \in \{0.02, 0.05\}$ of uniformly random rewrites per tick.
  Criterion: ordered regimes should degrade gracefully; if motif utility collapses immediately, structure is fragile.

**Cost-proxy swaps.**

- Replace linear c_* proxy with enumerative MDL on checkpoints.
  Criterion: decisions should remain correlated ($\rho_{rank} \geq 0.6$); large drifts imply proxy hacking.

**Decision rule.** If motifs vanish or invert (negative utility) under these stresses while null controls hold, downgrade or reject the emergence claim.

**D. Negative transfer: motifs that fail portability invalidate emergence claims**

**Definition.** A promoted motif library Lib_train should reduce codelength and violation mass when enabled on fresh seeds not used in discovery.

**Protocol (AB on held-out seeds).**

- With vs. without Lib_train, same tick budget and schedules.
- Metrics (per seed):
  - $\Delta$MDL_xfer = L_MDL(noLib) − L_MDL(withLib)
  - $\Delta$coh_xfer = L_coh(noLib) − L_coh(withLib)
  - External micro-task gain: $\tau$ = score_withLib − score_noLib

**Acceptance.**

- Median $\Delta$MDL_xfer > 0 and $\Delta$coh_xfer > 0 with 95% CI excluding 0 (bootstrap over seeds).
- For at least half of top-10 motifs by utility: $\tau > 0$.

**Negative transfer (invalidate).**

- If median $\Delta$MDL_xfer ≤ 0 **or** $\Delta$coh_xfer ≤ 0, or if $\tau \leq 0$ for the majority of promoted motifs with CIs excluding positive, then the library **harms** generalization.
- Retrospectively demote offending motifs; if the effect persists after curation, the central emergence claim is **invalidated**.

**Practical red flags (actionable)**

- **Motif inflation:** library grows but L_MDL plateaus; check save(M) accounting and require minimum usecount.
- **Witness churn:** high attach/detach rates; track conf decay—if frequent, witnesses are unstable.
- **Limit cycles:** periodic oscillation in J without net ΔL_MDL; detect via spectral peak in ΔJ_t.
- **Hyperparameter fragility:** tiny changes in λ, γ, ε flip outcomes; report sensitivity curves and require wide stable bands.

**Pre-registration & reporting**

- Pre-register thresholds (r*, δ_MDL*, α, N, K) and stress protocol before runs.
- Publish **run ledgers**, motif libraries (with hashes), and **counterexample suites**.
- Declare up front: if falsification criteria are met, we will report **failure** and the most plausible cause (collapse vs noise vs artifact), not re-tune until success without disclosure.

This section ensures ATSECTHI is an empirical claim—easy to disconfirm. If recurrent, compressive, law-stabilizing motifs **do not** survive seeds, shuffles, and stresses **and** they fail to transfer, we should conclude that coherence-seeking rewrites (as configured) do **not** yield emergent intelligence.

## VIII. Instrumentation & Reproducibility

### A. Deterministic seeds, manifest logging, and run ledgers

**Deterministic seeding (multi-stream).**
Use one master seed per run and derive independent streams for each stochastic subsystem (proposal policy, predictor dropout, adversary, curriculum gates).

- Stream derivation (64-bit): seed_stream = Hash64(master_seed ‖ stream_name)
- Streams: {rng_policy, rng_predictor, rng_adversary, rng_curriculum, rng_bootstrap}
  Pin any nondeterministic library flags (e.g., BLAS, GPU kernels) to deterministic modes.

**Rewrite manifest (append-only, event-sourced).**
One line per *accepted* rewrite; sufficient to reconstruct the run exactly.

json

CopyEdit


```
{
 "tick": 51215,
 "opcode": "compose",
 "args": {"g":"m_0091","f":"m_0042"},
 "pre_hash": {"U":"u:abcd...", "Lib":"lib:9876..."},
 "post_hash": {"U":"u:beef...", "Lib":"lib:9876..."},
 "Δ": {"L_MDL": -1.3, "L_coh": -0.2, "R": 0.0, "Δk": {"M_diamond": 0.41}},
 "ΔJ_hat": -1.5, "σ_hat": 0.3, "accepted": true
}
```


- pre_hash/post_hash are content hashes of the canonical universe and library (e.g., BLAKE3).
- Include predictor estimates and uncertainty to audit selection logic.

**Run ledger (human-readable index).**
A small YAML/JSON summarizing everything needed to replay or audit:

- run_id, master_seed, git_commit, build_fingerprint, hardware/OS/runtime
- Hyperparameters and schedules: (λ, γ, ε, T), cost vector c_*
- File pointers: manifest path, snapshot index, metrics CSV/Parquet, logs
- Integrity: signed digest of the ledger + manifest

**Version & provenance.**

- Record a software SBOM (SPDX) at start-up.
- Sign ledgers with an Ed25519 key; publish the public key.
- If a run is later invalidated, issue a **tombstone** file explaining why (bug, nondeterminism, hardware drift).

**B. Per-epoch universe diffs, invariant catalogs, and MDL audit trails**

**Canonical universe serialization.**

- Deterministic ordering of Obj, Mor, CompPairs, Ids, Functors, 2Cells.
- Namespacing: A_####, m_#### (zero-padded) to ensure lexicographic stability.
- Normalize(·) before every snapshot so structural equality implies byte-equality.

**Per-epoch diffs (compact).**
Store deltas against the previous epoch snapshot (add/remove/modify) with counts:
Δ|Obj|, Δ|Mor|, Δ|CompPairs|, Δ|Witness|, Δ|MotifRef|.

- Keep both **tick snapshots** (light) and **epoch snapshots** (full).
- Verify apply(patch(epoch−1)) → epoch under a unit test.

**Invariant (motif) catalog.**
For each promoted motif M:

- canon_id: canonical label (typed WL hashing or nauty canonical form)
- schema: minimal pattern (typed multigraph), arity, parameters
- expansion_map: how MotifRef(M) expands back to primitives
- witness_hooks: required 2-cells/paths for safe contraction
- Running stats: usecount, r, τ, μ^MDL, μ^coh, conf, last_seen_tick
- Hash of the schema+hooks to anchor references in manifests

**MDL audit trail (balance sheet).**
Log a full breakdown per epoch:

json

CopyEdit


```
{
  "epoch": 1024,
  "L": {"L_MDL": 102345.7, "L_sym": 33100, "L_rules": 9200,
        "L_motiflib": 4800, "L_witness": 6100, "L_deriv": 62500,
        "L_reuse": 1354.3},
```

"coh": {"E_assoc": 18, "E_id": 0, "E_fun": 3, "E_nat": 2},

"check": {"ΔJ": -1.62, "ΔL_MDL": -1.30, "λΔL_coh": -0.12, "∑βΔk": 0.20, "γΔR": -0.40,

"residual": -0.00}

}

**Energy check per tick:**
residual = ΔJ_measured − (ΔL_MDL + λ ΔL_coh − ∑_k β_k Δ_k + γ ΔR)
Require |residual| ≤ τ_resid (proxy arithmetic can set τ_resid = 1e−9 with rational costs).

**Heavy MDL recalibration.**
On every Nth epoch, compute an enumerative codelength L̂_MDL and track drift:
δ = |L̂_MDL − L_MDL|.
If δ > δ*, recalibrate the proxy c_* and log the adjustment.

**C. Public replication kit and "red team" perturbation scripts**

**One-command reproduction.**
Provide a Docker/Apptainer image pinned to exact versions, plus a reproduce.sh that:

1. Verifies checksums of code and config.
2. Runs the specified run_id with master_seed and schedules.
3. Emits manifest, snapshots, metrics, and signatures to an artifacts/ folder.

**Environment pinning.**

- CPU/GPU determinism flags on; if GPU nondeterminism persists, fall back to CPU for predictor or turn on deterministic kernels (with performance note).
- Record torch/cudnn determinism toggles; export env vars in the ledger.

**Public artifacts.**

- Upload ledgers, manifests, epoch snapshots, motif catalogs, and metrics to an archive (e.g., Zenodo) and mint a DOI.
- Include SBOM, license, and the public verification key.

**Red-team suite (scripted).**
Packaged experiments anyone can run with the kit:

1. **Curriculum ablations:** --no-mdl, --no-coh, --no-witness, --no-kl.
    - Expected: motif densities and transfer degrade in characteristic ways.
2. **Law permutations:** swap stage order; unlock Functoriality early/late.
    - Expected: robust motifs persist within tolerance bands.
3. **Noisy rewrites:** inject $\nu \in \{0.02, 0.05\}$ random rewrites per tick.
    - Expected: graceful degradation, not collapse.
4. **Operator mix stress:** ±50% prior shifts over {add, compose, cancel_id, merge_parallel, quot_obj}.
    - Expected: smooth response, no brittle dependence.
5. **Proxy swap:** rerun evaluation with heavy $\hat{L}$_MDL; compare rank correlation on decisions.
    - Expected: $\rho$_rank ≥ 0.6.
6. **Counterfactual replay:** start from epoch E, splice alternative rewrite subsequences; compare L_MDL, L_coh, μ.
    - Expected: motifs that claim causality should change outcomes predictably.

**Replay engine.**

- Reads the ledger + manifest, reconstructs states deterministically.
- Supports **branching replays** by replacing a manifest slice [t1,t2) with an alternative; writes a *derivative run ledger* with a new run_id chained to the parent via parent_hash.

**CI checks (small budgets).**

- Nightly: 4 seeds × 1e4 ticks on CPU; assert reproducibility (byte-equal hashes), pass/fail on invariants (e.g., $\rho_{\diamond} > 0$), and residual tolerance.
- On PRs: run the red-team *Noisy rewrites* test with $\nu$=0.02; fail build if motif utility collapses (util(M)_median ≤ 0).

**Minimal folder layout (published).**

```
pgsql
CopyEdit
artifacts/
  run_<id>/
    ledger.json
    manifest.jsonl
    snapshots/
      epoch_000000/ (full)
      tick_00000123.json (light)
    metrics.parquet
    motifs_catalog.json
    signatures/
      ledger.sig
      manifest.sig
    sbom.spdx.json
README.md
reproduce.sh
redteam/
  ablate_no_mdl.sh
  permute_curriculum.sh
  inject_noise.sh
  stress_opmix.sh
  proxy_swap_eval.sh
```

**What counts as a faithful replication?**

- Matching post_hash.U and post_hash.Lib at each epoch (or tick, if requested).

- Identical motif catalogs (IDs + schemas + stats within tolerance).
- Metric parity within tolerances:
  $|L_{MDL} - L_{MDL_ref}| \le 1e{-}6$, $|L_{coh} - L_{coh_ref}| \le 1e{-}6$, $|\mu - \mu_{ref}| \le 1e{-}6$.

**Public checklist (per paper release).**

1. Seeds, schedules, and hyperparameters disclosed.
2. Ledgers + manifests + signatures uploaded with DOI.
3. Reproduction script yields byte-identical hashes on two independent machines (documented).
4. Red-team suite run reports attached, including any failures and mitigations.

---

This instrumentation makes ATSECTHI auditable end-to-end: every accepted rewrite is reproducible; every MDL claim has an itemized receipt; every promoted motif is cataloged with hashes and utility; and anyone can re-run (or red-team) the entire process from a single command.

**IX. Case Studies (Toy → Non-Toy)**

**A. Toy: rapid emergence of commutative triangles and diamond squares**

**Setup.**
Seeds: N = 64; ticks per run: 2·10^5. Stages 0–2 only (no functors, no 2-cells beyond path witnesses). Costs: c_o=6, c_m=8, c_c=4, c_id=2. Schedules: moderate exploration early, anneal temperature. Null baselines: label-shuffle and degree-preserving rewires.

**Protocol.**

1. Bootstrap with $|Obj| \sim Uniform\{3,\ldots,8\}$ and sparse Mor.
2. Enable {compose, cancel_id, assoc_nf, close_square, merge_parallel}.
3. After each tick: Normalize, detect motifs, update L_MDL, L_coh.

**Key measurements.**

- Triangle density: $\rho_{\triangle} = C_{\triangle} / |Mor|$
- Diamond density: $\rho_{\diamond} = C_{\diamond} / |Mor|$

- Recurrence across seeds: r_M = |M_seed ∩ M_ref| / |M_ref|
- Compression per event: ΔL = L_MDL(before) − L_MDL(after)

**Representative outcome (illustrative).**

- Early regime (t ≤ 3·10^4): sharp rise in ρ_△ from ≈ 0 to ≈ 0.12, ρ_◇ to ≈ 0.07.
- Bursty compression aligned to motif promotions: median ΔL per burst ≈ 11.4.
- Recurrence: triangles r_△ ≈ 0.78, diamonds r_◇ ≈ 0.71.
- Null gap: z_△ ≥ 6.0, z_◇ ≥ 4.5.

**Micro-walkthrough (one triangle).**
Objects (A,B,C) with f: A→B, g: B→C.

1. compose(g,f) → h := g∘f: A→C.
2. close_square/triangle proposes u: A→C.
3. Normalization yields NF(h)=h. Attach witness ω: Path(u, h).
4. merge_parallel(u,h) fires:
   ΔL_MDL ≈ +len(u) − (overhead + reuse(h)), typically positive; ΔL_coh ≤ 0.
   Promote **triangle** motif when util(△) ≥ θ.

**Takeaway.**
Under minimal priors, **commutative triangles and diamonds** appear rapidly, recur across seeds, and coincide with **discrete MDL drops**, supporting the claim that lawful motifs are favored by coherence-seeking compression.

---

**B. Mid-scale: learned functorial lifts across sub-universes**

**Setup.**
Seeds: N = 16; ticks: 5·10^5. Unlock Stages 3–4 (functoriality and 2-cell witnesses). Create two sub-universes per run ($\mathcal{C}$_1, $\mathcal{C}$_2) with limited overlap; allow optional F: $\mathcal{C}$_1 → $\mathcal{C}$_2.

**Protocol.**

1. Discover motifs in $\mathcal{C}$_1 (triangles/diamonds).
2. Unlock add_functor, initialize F on a small anchor set.
3. Penalize functor violations in L_coh:
   E_fun contributes to L_coh = α_assoc·E_assoc + α_id·E_id + α_fun·E_fun + …

4. Track whether motifs “lift” under F: given M ⊂ $\mathcal{C}$_1, detect F(M) ⊂ $\mathcal{C}$_2 and measure utility.

**Metrics.**

- Functorial error rate: err_F = E_fun / |checks_F|
- Lift utility: util_F(M) = E[ΔL_MDL | F(M)] + ρ·E[−ΔL_coh | F(M)]
- Transfer gain within the run:
  Δcoh_F = L_coh(noF) − L_coh(withF) on $\mathcal{C}$_2 windows
- Cross-seed portability (AB):
  ΔMDL_xfer = L_MDL(noLib) − L_MDL(withLib)

**Representative outcome (illustrative).**

- err_F decays from ≈ 0.15 to ≤ 1e−3 over 1.5·10^5 ticks.
- For top motifs M, median util_F(M) > 0; naturality squares emerge as high-utility lifts.
- In $\mathcal{C}$_2, enabling F-lifted motifs yields Δcoh_F ≈ 12–18% drop and ΔMDL_xfer > 0 in matched budgets.

**Example (functorial check).**
Given f: A→B, g: B→C in $\mathcal{C}$_1,
F(g ∘ f) ?= F(g) ∘ F(f) in $\mathcal{C}$_2.
When a missing edge in $\mathcal{C}$_2 is proposed and certified by witnesses,
ΔL_coh < 0 (repairs E_fun) and, after reuse, ΔL_MDL > 0.

**Takeaway.**
With functoriality active, ATSECTHI **learns lifts**: motifs discovered in one sub-universe compress and stabilize another via F, evidencing **structured transfer** without external data.

---

**C. HoTT probe: path equivalences that reduce MDL in higher cells**

**Setup.**
Same mid-scale runs, with the **lightweight univalence-style probe** enabled. Probes suggest cand_equiv(A,B) scored by structural similarity; acceptance requires **transport maps** and **measured gains**.

**Acceptance test.**
Promote A ≃ B only if:

- There exist transports tr_{A→B}, tr_{B→A} over incident hom-structure,
- ΔL_MDL > 0 ∧ ΔL_coh ≤ 0 after quotienting and re-wiring,
- No new violations introduced; witnesses (2-cells) certify merged parallels.

**Higher-cell handling.**

- When parallel composites coincide post-quotient, attach path witnesses ω: Path(u,v).
- If witnesses themselves recur, record higher paths (chains), contributing to ρ_2+.

**Metrics.**

- Equivalence acceptance rate: acc_≃ = accepted / proposed
- Post-quotient savings: ΔL_MDL^≃ per accepted equivalence
- Higher-path density: ρ_2+ = C_2+ / |Mor|
- Conflict rate (rollback): conflict = retires / accepted (should be low)

**Representative outcome (illustrative).**

- acc_≃ ≈ 0.12 after early exploration; rises to ≈ 0.28 as the predictor matures.
- Median ΔL_MDL^≃ ≈ 9.3 proxy bits per accepted equivalence; ΔL_coh ≤ 0.
- ρ_2+ increases modestly (≈ 0.01 → 0.035), reflecting stable 2-cell scaffolding.
- Conflicts rare (< 3%) and rollbacks restore invariants without lasting damage.

**Example (object quotient).**
Probe proposes A ≃ B. Transports remap incident morphisms; duplicates u,v: A→C collapse under a witness ω, enabling merge_parallel(u,v) and reducing both codelength and violation mass:
ΔL = L_MDL(before) − L_MDL(after) > 0, ΔL_coh ≤ 0.

**Takeaway.**
Localized, testable **equivalences** (in the HoTT sense) can be accepted when they **pay rent**—they must **compress** and **not increase violations**—yielding modest but reliable higher-cell structure that further stabilizes contractions.

---

**What these case studies collectively show**

1. **Toy** universes already exhibit **lawful motifs** that **recur** and **compress**.
2. **Mid-scale** runs demonstrate **structured transfer** via **functorial lifts**.
3. **HoTT probes** allow **equivalence-driven refactors** guarded by 2-cells, adding a controlled higher-dimensional layer that continues to **reduce MDL** without destabilizing coherence.

These staged results support the central thesis: **coherence-seeking rewrites**—without external data—discover **reusable structure** that generalizes within the categorical aether.

**X. External Benchmarks (Zero-Data Transfer)**

**Principle.** Benchmarks are **held-out** micro-tasks never used in discovery. ATSECTHI faces them with its **frozen** motif library Lib_train (learned inside the aether) and **no new external data**. We report AB results: with vs without Lib_train, same tick budgets.

**Common metrics.**

- Accuracy: Acc = solved / total
- Time-to-solve (ticks): τ_solve
- Proof/rewrite length: |S| (accepted rewrites to completion)
- Residual violation mass at solve: L_coh^*
- Codelength delta vs. start: ΔMDL^* = L_MDL(start) − L_MDL(solve)
- Transfer gains: ΔAcc = Acc_with − Acc_without, Δτ = τ_without − τ_with

**Baselines.** (i) Random proposals under KL control, (ii) MDL-only (λ=0), (iii) No-witness. ATSECTHI must beat all three.

---

**A. Diagram chasing and small-category reasoning tasks**

**Task family.** Synthetic finite categories with hidden commuting constraints. Inputs are typed graphs; queries ask for missing arrows or equalities that make specified diagrams commute.

1. **Fill-the-triangle.**
   Given A→^f B→^g C, find h: A→C such that h = g∘f.
   Score solve if the system produces h and attaches a witness ω: Path(h, g∘f) with L_coh unchanged or reduced.

2. **Complete-the-square (diamond).**
   Given A→^f B→^h D and A→^g C→^k D, but one edge missing, propose the missing edge and certify h∘f = k∘g.
   Requirement: NF(h∘f) = NF(k∘g) and witness attached.

3. **Parallel-path equality.**
   Given two length-≤4 composites u,v: X→Y, decide equality in normal form and, if equal, compress via merge_parallel.
   Success if u=v is established and ΔMDL^* > 0.

4. **Functorial lift check.**
   Two small categories $\mathcal{C}$, $\mathcal{D}$ with partial F: $\mathcal{C}\to\mathcal{D}$. Verify F(g∘f) = F(g)∘F(f); propose missing component if needed.
   Score only if E_fun decreases: ΔL_coh^fun < 0.

**Instance generation.**

- Sizes: |Obj| ∈ {4,6,8}, edge density p ∈ [0.2,0.35].
- Hidden constraints sampled from a motif grammar (triangles, squares) not exposed to the agent.
- Each query packs: seed, spec IDs, and a solve budget (ticks).

**Metrics (per batch).**

- Acc_△, Acc_◇, Acc_‖, Acc_F and corresponding τ_solve, |S|, ΔMDL^*.
- Portability statistics: recurrence of library motifs during solves r_solve = |hits(Lib_train)| / |expected|.

**Zero-data transfer claim.** Gains must come from **reuse**: ΔAcc > 0 ∧ Δτ > 0 with Lib_train frozen; no task-tuned learning.

---

**B. Rewriting puzzles (Knuth–Bendix style) without exposure to text**

**Task family.** Finite term-rewrite systems (TRSs) on a tiny signature, compiled into categorical form (objects = sorts, morphisms = terms, composition = substitution). ATSECTHI performs completion-like steps using its rewrite operators.

1. **Normal-form reachability.**
   Given term t, reductions R, decide if NF(t) equals a target t*.
   Success if NF(t)=t* via β-style normalization and contractions; report |S| and ΔMDL^*.

2. **Critical pair convergence.**
   Present a critical pair (u,v); require building witnesses to show joinability u ↘ w ← v in NF.
   Success if witnesses certify equality and L_coh does not increase.

3. **Local confluence patch.**
   Provide a small non-confluent TRS. System must propose **one** additional oriented rule (encoded as a motified contraction) that makes all observed critical pairs join.
   Score if observed_nonjoinable → 0 and ΔMDL^* > 0.

4. **Equational theory compression.**
   Start from redundant equations; compress to a smaller, equivalent set by mergers and motif reuse.
   Success threshold: L_MDL(reduced) ≤ L_MDL(baseline) − δ, with δ > 0.

**Construction.** Random TRSs with bounded left/right sizes, designed so solutions correspond to small categorical motifs (triangles, diamonds, associators). No natural-language exposure.

**Metrics.**

- Solve rate, average rewrite count, witness attachment rate, post-solve L_coh^*.
- Heavy-check consistency: recompute enumerative codelength L̂_MDL for the reduced theory; require |L̂_MDL − L_MDL| ≤ δ*.

**Transfer signal.** With Lib_train, expect higher witness rates and fewer steps: Prec@k for chosen rewrites rises, |S| drops.

---

**C. Program-synthesis micro-problems framed as categorical rewrites**

**Idea.** Treat tiny synthesis tasks as **diagram completion** under specifications—no tokens, just arrows and commutativity.

1. **Spec-as-commuting diagram.**
   Given objects I (inputs), O (outputs), and a library of primitive morphisms {p_j}, synthesize a composite f: I→O so that designated test squares commute (behave identically on provided projections).
   Solve if ATSECTHI constructs f with witnesses showing required equalities.

2. **Monoid homomorphism builder.**
   Two small monoids (M, ⊙, e_M), (N, ⊗, e_N) encoded as single-object categories. Find

h: M→N such that h(x ⊙ y) = h(x) ⊗ h(y) and h(e_M) = e_N.
Success if functorial constraints are satisfied and E_fun → 0.

3. **Dataflow patching (typed).**
   A partial pipeline graph with a type mismatch is provided. Synthesize an adapter morphism (or factorization through an intermediate object) to restore end-to-end commutativity.
   Score if violations decrease: ΔL_coh < 0 and ΔMDL^* ≥ 0.

4. **Functor lift of a construction.**
   Given F: $\mathcal{C}\to\mathcal{D}$ and a composite g∘f in $\mathcal{C}$, synthesize F(g∘f) in $\mathcal{D}$ using available pieces, ensuring F(g∘f) = F(g)∘F(f).
   Solve if the missing arrow is created, witnesses attached, and E_fun drops.

**Specs and guards.**

- Specs are **purely structural**: commutativity constraints and typing; no text, no examples beyond arrow-level checks.
- Budgets are fixed (ticks, operator counts); Lib_train is frozen; only reuse is allowed.

**Metrics.**

- Acc_synth, mean |S|, mean τ_solve, ΔMDL^*, L_coh^*
- Motif activation fraction during solves: ϕ = used_MotifRef / total_steps
- AB transfer: ΔAcc, Δτ, Δ|S| with 95% CI (bootstrap over instances)

---

**Benchmark protocol and acceptance**

**Zero-data transfer protocol.**

1. Train-discover motifs internally on a separate seed set. Freeze Lib_train.
2. Instantiate benchmark instances with independent seeds and hidden constraints.
3. Run ATSECTHI with **no parameter learning**—only rewriting and reuse.
4. Compare to baselines under identical KL and temperature schedules.

**Acceptance thresholds (example).**

- Diagram chasing: ΔAcc ≥ 0.15 and Δτ ≥ 10% reductions vs no-library baseline.
- Rewriting puzzles: ≥ 0.10 absolute gain in solve rate, |S| reduced by ≥ 15%.

- Program synthesis: Acc_synth ≥ 0.6 overall and ΔAcc ≥ 0.12 vs baselines.

**Statistical tests.**

- Proportion differences: Wilson intervals; test ΔAcc > 0 at α=0.01 with BH correction across task families.
- Time/length: Mann–Whitney U; report effect sizes (Cliff's δ).
- Robustness: re-run with v=0.02 noisy rewrites; gains should persist (degrade by ≤ 30%).

**Leakage checks.**

- Verify Lib_train motifs are **generic** (triangles, diamonds, witnesses) not hand-crafted for instances.
- Hash and disclose benchmark generators and seeds; third parties must reproduce scores with the public kit.

**Primary success claim.**
If Lib_train—learned **without** any benchmark exposure—consistently improves solve rates, reduces steps, and lowers violation mass across all three families, then ATSECTHI demonstrates **zero-data transfer**: reusable, lawful structure formed in the categorical aether confers **out-of-domain utility** without text or labels.

## XI. Safety, Alignment, and Sandboxing

### A. Local-truth containment: domain-bounded coherence prevents overreach

**Principle.** ATSECTHI's "truths" are **local** to its categorical universe. They are validated only by typed rewrites, normalization, and witnesses inside the sandbox.

**Containment mechanisms.**

- **Closed world:** no external corpora, no network I/O, no system calls beyond a whitelisted filesystem path for ledgers/artifacts.
- **Typed legality:** every action must satisfy typing and composability; illegal proposals are discarded pre-score.
- **Coherence gate:** rewrites that increase violation mass are disfavored unless exploration budget explicitly permits (and only transiently).
  Guard: commit(S) ⇒ ΔL_coh(S) ≤ δ_coh_allow with δ_coh_allow ≤ 0 outside explore phases.

- **Local-truth labeling:** every promoted result is tagged with its **scope** (category IDs, stage, cost vector), preventing re-interpretation outside domain.

**Policy alignment via incentives.**

- Objective aligns with containment: $J = L_{MDL} + \lambda L_{coh} - \sum_{k} \beta_k \Delta_k + \gamma R$.
  Lower J requires shorter codes, fewer violations, and verified motifs; there is **no reward** for external effect or semantic claims.
- **Conservative exploration:** KL budget caps step-to-step drift: $KL(p_t \| p_{t+1}) \le \varepsilon$. This prevents runaway proposal policies.

**Resource bounds.**

- Fixed quotas per run: ticks $\le T_{max}$, $|Obj| \le O_{max}$, $|Mor| \le M_{max}$, wall-clock guard.
- If bounds are approached, growth rewrites are throttled and contractions prioritized.

**B. Interpretability: every step is a typed rewrite with witnesses**

**Event sourcing for legibility.**

- **Manifests** record each accepted rewrite with pre/post hashes, predicted and realized deltas, and legality verdicts.
- **Universe diffs** show adds/removes/modifies for Obj, Mor, Comp, 2Cells, MotifRef.

**Proof-carrying contractions.**

- Parallel-path merges require an attached 2-cell witness $\omega$: Path(u, v); object quotients require an equivalence bundle with transport maps.
- **No witness ⇒ no contraction.** This keeps equalities auditable and reversible.

**Causal traces.**

- Each motif promotion references the **evidence set** (seeds, ticks, deltas) that justified it.
- A **witness DAG** links contractions to the witnesses that enabled them; replay validates every edge.

**Human-readable invariants.**

- The invariant catalog stores canonical schemas and expansion maps; inspectors can expand any MotifRef to primitives to see **exactly** what was compressed.
- MDL audit trails function as itemized receipts: who saved what, when, and why.

**Predictor transparency.**

- The self-predictor logs ΔJ_hat and uncertainty σ_hat; selection uses UCB = −ΔJ_hat − u·σ_hat.
- **Ablation replays** (predictor off) quantify the predictor's true contribution to decisions.

**C. Kill-switches: invariants only accrue with cross-seed verification**

**Two-phase motif governance.**

- **Shadow phase (no credit).** A candidate motif M_k is detected and tracked, but **β_k = 0** (no reward) until cross-seed checks pass.
- **Promotion phase (credit on).** Only after recurrence and utility are verified across independent seeds does M_k earn weight.

**Formal gate (per motif).**

- Accrual switch:
  accrue_k = 1 if (r_k ≥ r* ∧ μ_k^MDL ≥ δ_MDL* ∧ μ_k^coh ≥ 0 ∧ conf_k ≥ c*) else 0
- Where:
  r_k = |M_k(seed) ∩ M_k(ref)| / |M_k(ref)| (recurrence)
  μ_k^MDL = E[ΔL_MDL | apply(M_k)] (compression utility)
  μ_k^coh = E[−ΔL_coh | apply(M_k)] (repair utility)

**Automatic demotion and quarantine.**

- **Negative transfer:** if on held-out seeds ΔMDL_xfer ≤ 0 or Δcoh_xfer ≤ 0, set β_k ← 0, tag M_k as quarantined, and block its reuse.
- **Conflict-triggered rollback:** if a promoted witness later conflicts (counterexample), decrement conf; below c_quarantine, trigger rollback to the last snapshot where dependent merges are not yet committed.

**Run-level kill-switches.**

- **Stagnation kill:** if ⟨ΔL_MDL⟩_W ≈ 0 ∧ ⟨ΔL_coh⟩_W ≈ 0 ∧ μ ≈ 0 over W, pause proposals, dump diagnostics, and require operator approval to resume.
- **Divergence kill:** if κ = L_MDL/|Mor| exceeds a high watermark and ϵ = L_coh/|CompPairs| rises, force a **compress-only** mode (growth ops disabled) or terminate.

- **Entropy kill:** if H(p_{t+1}) collapses below a floor (policy collapse), reset exploration parameters to safe defaults or freeze learning.

**Human-in-the-loop safeties (optional).**

- **Two-person rule for equivalences:** object-level quotients (A ≃ B) require a second, independent run to reproduce gains before permanence.
- **Red-team gate:** promotion blocked until the red-team perturbation scripts report stable utility under noise and curriculum permutations.

**Sandbox perimeter.**

- **I/O firewall:** motif libraries, manifests, and snapshots are the only outputs; no code generation is executed outside the sandbox.
- **Determinism lock:** replay must reproduce hashes; if not, runs are invalidated and their artifacts are marked with tombstones.

---

**Safety summary.**
Containment is enforced by **typed legality** and **coherence costs**; interpretability is guaranteed by **event-sourced rewrites** and **witnessed contractions**; and power accrues to invariants **only** after **cross-seed verification** with automatic demotion on negative transfer. Together these make ATSECTHI auditable, rollbackable, and resistant to overreach—all while remaining productive inside its strictly local domain.

## XII. Related Work

### A. Inductive logic programming, neural theorem proving, symbolic regression

**Inductive Logic Programming (ILP).**
Classical ILP induces logic programs from examples under a fixed language bias (predicates, modes) and background knowledge. Strengths: declarative hypotheses, strong generalization when the bias matches the task. Limits relative to ATSECTHI: (i) depends on labeled examples and negatives; (ii) hypothesis search is goal-directed toward explaining data, not toward compressing an evolving *universe*; (iii) axioms don't self-organize unless engineered. ATSECTHI is **example-free** and **ecological**: it grows a world, then adopts hypotheses (motifs/witnesses) only if they reduce L_MDL and L_coh.

**Neural(-guided) Theorem Proving (NTP / ATP).**
Neural guidance for proof search (e.g., premise selection, tactic prediction) boosts success rates in fixed axiom systems. Compared to ATSECTHI: (i) proof targets are externally specified; (ii) success = find a derivation; (iii) axioms are static (or curated offline). ATSECTHI inverts this: there is no external theorem; success = sustained **compression with lawfulness**. It **creates** and **retires** its own lemmas (motifs) by measured utility across seeds.

**Symbolic Regression (SR).**
SR discovers compact equations that fit numeric data (genetic programming, sparse regression, differentiable SR). It is compressive but **data-driven** and numeric. ATSECTHI is **structure-driven** and discrete: it compresses typed graphs and rewrite histories. Where SR minimizes a prediction loss plus simplicity, ATSECTHI minimizes:
$J = L_{MDL} + \lambda L_{coh} - \sum_{k} \beta_k \Delta_k + \gamma R$
with $L_{MDL}$ over categorical grammars, not real-valued residuals.

**Summary (A).**
ILP/NTP/SR all rely on **external tasks or datasets**. ATSECTHI asks: can **law-seeking dynamics** alone—compression + coherence—yield reusable invariants without any examples?

---

**B. Self-play emergence (AlphaZero-class) vs. law-seeking emergence (ATSECTHI)**

**AlphaZero-class self-play.**

- Environment: fixed game rules; reward: extrinsic win/loss.
- Policy/Value networks trained by Monte Carlo search; emergence = strategies that maximize external score.
- Knowledge storage: opaque parameters; interpretability via post-hoc analysis.

**ATSECTHI law-seeking.**

- Environment: *endogenous* categorical “aether”; no opponent, no extrinsic reward.
- Objective is **intrinsic**: shorten description ($L_{MDL} \downarrow$), reduce violations ($L_{coh} \downarrow$), reuse verified invariants ($\Delta_k \uparrow$), maintain healthy exploration ($R$).
- Knowledge storage: explicit **motif libraries** and **witnesses**; every contraction carries a certificate.

**Key contrasts.**

- **Target:** win-rate vs. ΔL_MDL with coherence.
- **Learning signal:** sparse extrinsic vs. dense intrinsic (per rewrite).
- **Generalization vehicle:** policy weights vs. portable **typed motifs**.
- **Safety/interpretability:** post-hoc vs. **event-sourced** proofs.

**Convergence intuition.**
Self-play climbs an external gradient; ATSECTHI climbs a **compression–coherence** gradient. Both can show phases and sudden jumps, but for ATSECTHI the "victories" are **discrete refactors** (compression bursts) that persist across seeds.

---

**C. Algorithmic information theory and compressive discovery**

**Kolmogorov complexity and MDL.**
AIT defines ideal description length K(x); MDL provides computable surrogates. ATSECTHI operationalizes this by maintaining a **code** for its universe and rewarding steps that shorten it:
$\Delta L_{MDL} = L_{MDL}(\text{before}) - L_{MDL}(\text{after}) \geq 0$
accepted when compatible with coherence:
$\Delta J = \Delta L_{MDL} + \lambda \Delta L_{coh} - \sum_{k} \beta_k \Delta_k + \gamma \Delta R < 0$

**Compression progress as intrinsic motivation.**
The system's intrinsic "reward" can be read as:
$r_t^{intrinsic} = \Delta L_{MDL} - \lambda (-\Delta L_{coh}) + \sum_{k} \beta_k \Delta_k - \gamma \Delta R$
(positive when codes shorten, violations drop, and verified invariants are reused under a healthy exploration budget). Unlike curiosity on prediction error, the signal is **proof-carrying**: savings must be auditably realized in the grammar.

**From patterns to invariants.**
AIT says regularities enable shorter codes. ATSECTHI refines this into **typed** regularities (motifs) with **witnesses**, turning raw repetition into **lawful** compression. Portability is the acid test: only motifs with cross-seed recurrence and positive transfer retain weight.

**Relation to compressive scientific discovery.**
The stance echoes "theories as compressors": better theories yield shorter encodings of data. The twist here is that **both the "theory" and the "data" are endogenous**: the universe is what the system builds; a good theory is one that *continues* to shorten it while keeping laws satisfied.

**Why not pure MDL?**
Pure compression would risk degenerate tricks (e.g., remove structure). The coherence term and witness requirements prevent that: contractions must respect typed equality, and:
L_coh(after) ≤ L_coh(before)
for most steps, with exploration only temporarily relaxing the constraint.

**Takeaway (C).**
ATSECTHI is a **computable AIT lab**: it measures discovery as **auditable drops in description length** gated by **formal coherence**, converting compression progress into reusable, verified invariants rather than opaque parameter changes.

## XIII. Roadmap (6–12 Months)

### Q1 (Months 0–3): Stabilize MDL gate, self-predictor, KL-budget; publish instrumentation

**Engineering goals**

- **MDL proxy v1.** Implement linear cost model with heavy-check recalibration every N epochs; expose c_* in config; log ΔL_MDL per tick.
- **Self-predictor v1.** Typed-graph encoder + multihead regression for ΔL_MDL, ΔL_coh, ΔJ, legality, uncertainty; UCB selection: UCB = −ΔJ_hat − u·σ_hat.
- **KL controller v1.** Trust-region update enforcing KL(p_t ‖ p_{t+1}) ≤ ε; entropy floor + ε-greedy fallback.
- **Normalization v1.** Global right-association; cancel_id; deterministic naming; byte-stable serialization.
- **Instrumentation v1.** Deterministic seeds (multi-stream), append-only manifest, epoch snapshots, MDL audit trail, residual check:
  residual = ΔJ − (ΔL_MDL + λ ΔL_coh − ∑ β_k Δ_k + γ ΔR) (require |residual| ≤ τ_resid).

**Experiments**

- 32 seeds × 2·10^5 ticks, Stages 0–2 only (no functors/2-cells).
- Schedules: exploratory early (γ, ε, T high), anneal; λ ramp.

**Success criteria (Q1 exit)**

- **Predictor actionability:** ρ_rank(−ΔJ_hat, −ΔJ) ≥ 0.6 and Prec@10 ≥ 0.6 in Q2 complexity quartiles.

- **Compression bursts:** heavy-tail in burst sizes with $\alpha \in [1.5, 3]$.
- **Motif emergence:** nonzero $\rho_{\triangle}$, $\rho_{\diamond}$ with $z \geq 3$ vs null shuffles.
- **Reproducibility:** byte-identical replays on 2 machines; residual tolerance met.

**Deliverables**

- Public **Instrumentation Report v1** (methods, schemas, hashes).
- Minimal **reproduce.sh** + container image; small red-team preview (noisy rewrites).

---

**Q2 (Months 3–6): Large seed-ensemble study; phase diagrams and motif statistics**

**Engineering goals**

- **Phase-scan harness.** Grid search over ($T$, $\varepsilon$, $\lambda$); automated runs; aggregation dashboards for ($\kappa$, $\epsilon$, $\mu$, EOC).
- **Registrar v1.** Canonical motif detection, utility estimation, promotion/demotion; $L_{reuse}$ wiring.
- **Null models.** Label-shuffle + degree-preserving rewire pipelines; z-scores.

**Experiments**

- 128 seeds × $3 \cdot 10^5$ ticks per ($T$, $\varepsilon$) grid point; $\lambda \in \{0.1, 0.3, 0.6\}$.
- Report **EOC ridge** where motif density and compression are maximal with controlled violations.

**Success criteria (Q2 exit)**

- **Motif recurrence:** for triangles/diamonds, median $r_M \geq 0.7$ in ≥ 60% seeds.
- **Phase structure:** clear separation of **noise / edge-of-chaos / order** by ($\mu$, $\epsilon$, $\kappa$, EOC) indicators.
- **Registrar utility:** promoted motifs yield $E[\Delta L_{MDL} \mid M] > 0$ and $E[-\Delta L_{coh} \mid M] \geq 0$.

**Deliverables**

- **Phase Diagram Preprint** (motif spectra, burst statistics, EOC maps).
- Expanded **red-team suite**: ablations (no-MDL / no-coh / no-KL), operator-mix stress, proxy-swap evaluation.

**Go/No-Go Gate A**

- If no parameter band yields r_M ≥ 0.6 with positive MDL gains, **halt scaling**; open a "failure note" and iterate on MDL proxy/predictor.

---

**Q3 (Months 6–9): Optional 2-cells as witnesses; measure associativity/identity repair rates**

**Engineering goals**

- **2-cells/witnesses v1.** Path witnesses ω: Path(u, v); attach on equality in NF or learned certificate under strict gates; conflict ledger + rollback.
- **HoTT probe (light).** probe(A,B) → cand_equiv(A≃B) with transport checks and acceptance: require ΔL_MDL > 0 ∧ ΔL_coh ≤ 0.
- **Functoriality unlock.** Stage 3 gates; E_fun penalties; add_functor operator with tiny anchor sets.

**Experiments**

- 16 seeds × 5·10^5 ticks with Stages 3–4 enabled.
- **Repair measurements:** time-to-fix under adversarial generators: τ_fix^assoc, τ_fix^id, τ_fix^fun (90% reduction criterion).
- **Witness stability:** conflict rate < 5%, quarantine/rollback functioning.

**Success criteria (Q3 exit)**

- **Repair rates:** median τ_fix finite and improving across stages; E_assoc, E_id, E_fun → near zero.
- **Equivalences:** acceptance rate acc_≃ ≥ 0.2 with median ΔL_MDL^≃ > 5 proxy bits; conflict rollbacks rare (< 3%).
- **Transfer within run:** functor lifts reduce L_coh in target sub-universes (Δcoh_F > 0) with matched budgets.

**Deliverables**

- **Witness & Repair Note** (assoc/identity/functor repairs, rollback logs).
- **Motif Catalog v1** with hashes, schemas, expansion maps, and utility stats.

**Go/No-Go Gate B**

- If witnesses fail (high conflict/rollback) or equivalence acceptance yields $\Delta L_MDL^{\simeq} \leq 0$, **freeze 2-cells** and proceed with first-tier motifs only.

---

**Q4 (Months 9–12): Replication package; position paper + empirical preprint**

**Engineering goals**

- **Replication kit v1.0.** Docker/Apptainer image; signed run ledgers; replay engine with branching replays; CI for nightly reproducibility.
- **Benchmark harness.** Zero-data transfer suites (diagram chasing, TRS puzzles, program-synthesis micro-tasks) with public seeds.

**Experiments**

- **Cross-universe transfer.** Train Lib_train on S_train; AB on S_test (frozen library).
- **External benchmarks.** Report ΔAcc, Δτ, Δ|S|, ΔMDL^* vs baselines (random, MDL-only, no-witness).

**Success criteria (Q4 exit)**

- **Transfer:** median $\Delta MDL_xfer > 0$ and $\Delta coh_xfer > 0$ with 95% CI excluding 0.
- **Benchmarks:**
    - Diagram chasing: $\Delta Acc \geq 0.15$, $\Delta\tau \geq 10\%$.
    - TRS puzzles: solve rate +0.10 absolute, steps −15%.
    - Program synthesis: $Acc_synth \geq 0.6$, $\Delta Acc \geq 0.12$.
- **Independent replication:** at least one external party reproduces hashes and headline metrics using the public kit.

**Deliverables**

- **Position Paper:** “Law-Seeking Emergence in a Data-Free Categorical Aether.”
- **Empirical Preprint:** ensemble study, phase diagrams, witness repairs, transfer benchmarks.
- Public archive (DOI) with: ledgers, manifests, snapshots, motif catalog, red-team scripts, SBOM, signatures.

**Cross-cutting tracks (all quarters)**

- **Safety & sandboxing.** Closed-world I/O, resource caps, kill-switches; two-person rule for object equivalences in Q3–Q4.
- **Pre-registration.** Fix thresholds (r*, δ_MDL*, α, N) and stress protocols before ensemble runs.
- **Risk log.** Track: proxy hacking, predictor overfit, limit cycles, hyperparameter brittleness; ship mitigations alongside results.
- **Compute hygiene.** Deterministic modes; fallback to CPU for predictor when GPU nondeterminism detected; nightly 4-seed CI health checks.

---

**Milestone checklist (condensed)**

- **M1 (end Q1):** MDL/predictor/KL stabilized; instrumentation report; reproducible 32-seed toy runs.
- **M2 (mid Q2):** Phase-diagram scans complete; motif recurrence with z-scores vs null.
- **M3 (end Q3):** Witnesses operational; measured repair rates; functor lifts with positive utility.
- **M4 (end Q4):** Public replication kit; external benchmarks show zero-data transfer gains; preprints released.

**Scale-up trigger (post-Q4):**
If Q4 criteria met, plan Q5 for larger universes, richer operator sets, and expanded HoTT fragments—otherwise, iterate on the failing gate (Go/No-Go A/B) before scaling.

## XIV. Minimal Claims, Maximum Testability

### A. We do not claim semantics or cognition—only emergent lawful structure

**Claim boundary.** ATSECTHI demonstrates that, inside a closed categorical sandbox, a rewrite process that optimizes **compression** and **coherence** produces **recurrent, reusable invariants** (motifs) with measurable utility. The objective is intrinsic:
$J = L_{MDL} + \lambda L_{coh} - \sum_{k} \beta_k \Delta_k + \gamma R$.

- **What we claim:** reproducible drops in L_MDL, sustained decreases in violation mass L_coh, and cross-seed motif recurrence with positive transfer on **structural** micro-tasks.

- **What we do not claim:** semantics, understanding, world models, language, goals, values, or general cognition beyond the sandbox. No external data are used; no conclusions are drawn about human meaning or real-world alignment.

**Interpretation scope.** All results are **local truths**—valid only within the encoded universe, its typing rules, and its instrumentation.

---

**B. Crisp failure modes and acceptance criteria**

**Primary success criteria (all must hold in at least one stable parameter band):**

1. **Recurrent invariants.** For ≥ 60% of seeds, top motif classes satisfy $r_M = |M_{seed} \cap M_{ref}| / |M_{ref}| \geq r^*$ with $r^* \in [0.6, 0.8]$, and show $z_M \geq 3$ vs null shuffles/rewires.
2. **Compression with lawfulness.** Windowed median $\Delta L_{MDL} > 0$ (bursty is fine) while $\Delta L_{coh} \leq 0$ on average (post-anneal). Heavy-check MDL correlates with proxy: rank correlation $\rho_{rank} \geq 0.6$.
3. **Transfer without new learning.** On held-out instances, frozen Lib_train yields $\Delta Acc > 0$, $\Delta\tau > 0$, and $\Delta MDL_{xfer} > 0$, with 95% CIs excluding 0.
4. **Predictor usefulness.** Compared to a no-predictor baseline under identical KL/temperature, Prec@10 ≥ 0.6 and $\Delta_{gain} = E[\Delta J_{with}] - E[\Delta J_{without}] > 0$.

**Hard falsifiers (any one suffices to reject for a configuration):**

- **No recurrence:** $r_M < 0.5$ for all motif types despite large seed/shuffle ensembles.
- **No compression:** $E[\Delta L_{MDL}] \leq 0$ beyond transient warm-up, or MDL proxy fails heavy-check ($\rho_{rank} < 0.4$).
- **Negative transfer:** median $\Delta MDL_{xfer} \leq 0$ or $\Delta coh_{xfer} \leq 0$ on held-out seeds.
- **Non-replicability:** manifests/snapshots do not reproduce byte-identically on two machines.
- **Hyperparameter fragility:** minimal success disappears with ±10% changes in (λ, ε, T) and operator mix.

**Stress test gates (must pass with graceful degradation):**

- Ablations (no-MDL / no-coh / no-KL / no-witness) behave as predicted; motif utility drops but diagnostic patterns match expectations.

- Noisy rewrites ($\nu \in \{0.02, 0.05\}$) reduce but do not annihilate transfer ($\Delta$Acc degrades by $\leq 30\%$).

All thresholds, seeds, and scripts are **pre-registered** and published; failure is reported as such without retrospective tuning to the same dataset.

---

**C. How results would justify scaling (or not)**

**When to scale (green-light):**

- A **stable band** in parameter space shows all four success criteria above, with stress tests passed and independent replication verified.
- Evidence of an **edge-of-chaos ridge** where motif density and compression peaks while L_coh stays low suggests room to grow capacity.

**Scaling path (graduated):**

1. **Capacity:** increase |Obj|, |Mor|, tick budgets, and operator repertoire incrementally; keep MDL heavy-checks frequent until proxy stability persists ($\rho$_rank $\geq 0.7$).
2. **Structure:** unlock functoriality, then 2-cells, then lightweight equivalence probes—each only after the prior tier meets transfer thresholds.
3. **Diversity:** run heterogeneous curricula and multi-sub-universe settings; verify portability across more varied graph topologies.
4. **Bench breadth:** expand zero-data benchmarks (diagram families, TRS suites, synthesis specs) while keeping the **frozen-library** rule.

**When *not* to scale (red-light):**

- Success hinges on a narrow hyperparameter spike or a specific curriculum order.
- Motifs inflate the library without sustained L_MDL gains (save(M) $\leq 0$ on median).
- Witness conflicts/rollbacks exceed a small ceiling (e.g., > 5%).
- Transfer turns negative or cannot be replicated externally.

**Decision heuristic.**
Scale **only** if the marginal gains from added structure (functors → 2-cells → probes) exceed the added complexity cost, measured by:
ROI_struct = ($\Delta$Acc + w_1 $\Delta\tau$ + w_2 $\Delta$MDL_xfer) − (cost_runtime + cost_instrumentation)
and ROI_struct > 0 across ≥ 2 independent ensembles.

**Bottom line.**
ATSECTHI's claims are intentionally minimal and aggressively testable: **recurrent, lawful compression** producing **reusable invariants** that transfer **without new data**. Clear quantitative gates, red-team stresses, and public replication ensure that either the claim **stands on evidence**—in which case scaling is justified—or it **fails cleanly**, in which case the path forward is to revise the objective, priors, or call the approach unproductive for the stated goal.

**References**


Youvan, D. C. (2025). *Aether-Born Intelligence: Emergence of Structure in Minimal-Information Categorical Graph Growth.* DOI: 10.13140/RG.2.2.18162.41922.

Youvan, D. C. (2025). *Beyond Intuition: Nontrivial Equivalences in Category Theory and Homotopy Type Theory.* DOI: 10.13140/RG.2.2.34159.75688.

Youvan, D. C. (2025). *Climbing the Gödel Boundary: GenesisBoard, Category-Theoretic Graph Growth, and the Iterative Topology of AGI Reasoning.* DOI: 10.13140/RG.2.2.27324.35208.

Youvan, D. C. (2025). *DIY Category Theory and Homotopy Type Theory on a PC: From Abstract Proofs to Verified Executable Code.* DOI: 10.13140/RG.2.2.12912.98563.

Youvan, D. C. (2025). *Emergence of Higher-Order Structure in Category-Theoretic Graph Growth: Experimental Evidence from the GenesisBoard Framework.* DOI: 10.13140/RG.2.2.21360.57606.

Youvan, D. C. (2025). *From Silicon to Structure: A Thought Experiment in the Evolution of Categorical Intelligence from a Consumer GPU.* DOI: 10.13140/RG.2.2.27396.44164.

Youvan, D. C. (2025). *From Tuples to Topoi: Rethinking Meaning, Metaphor, and Mind Through Category Theory.* DOI: 10.13140/RG.2.2.26440.84486.

Youvan, D. C. (2025). *Gödel, Escher, and Homotopy Type Theory: Structural Self-Similarity and the Birth of Categorical Consciousness.* DOI: 10.13140/RG.2.2.19763.68649.

Youvan, D. C. (2025). *Lattice-to-Hilbert Mapping in AI Systems: Toward a Homotopical Theory of Distributed Intelligence.* DOI: 10.13140/RG.2.2.29506.70089.

Youvan, D. C. (2025). *Local Truth in Categorical Simulations: GenesisBoard as a Laboratory for Domain-Bounded Coherence.* DOI: 10.13140/RG.2.2.11503.96164.

Youvan, D. C. (2025). *The Commutative Inferno: Descent into the Categorical Abyss.* DOI: 10.13140/RG.2.2.18896.42242.

Youvan, D. C. (2025). *The Oracle's Map: A Guided Journey Through Category Theory and Homotopy Type Theory.* DOI: 10.13140/RG.2.2.30640.47363.

Youvan, D. C. (2025). *Theophany as Topos: Category Theory at the Threshold of Divine Disclosure.* DOI: 10.13140/RG.2.2.15252.62085.

Youvan, D. C. (2025). *A Formal Reconstruction of Physics: Foundations in CT, HoTT, and Category-Driven Quantum Design.* DOI: 10.13140/RG.2.2.11511.06566.

Youvan, D. C. (2025). *Structural Cognition: Human and Artificial Intelligence in a Category Theory and Homotopy Type Theory World.* DOI: 10.13140/RG.2.2.19060.26246.

**Appendix A — core7.py: Minimal Aether Engine (CT/HoTT Toy)**

**Purpose.** core7.py is a compact sandbox that grows a tiny categorical "universe" (objects + morphisms), opportunistically closes triangles/squares/paths, and tracks a simple *compression* proxy. It's Windows-friendly, version-tagged, and ships a built-in analyzer.

**Universe representation (Section II).** The state is a typed dictionary with:

- objects, morphisms, two_cells, lemmas, plus epoch and run_dir. This is the minimal "aether" needed for objects, arrows, and higher cells at toy scale.
- Runs are placed under runs/run-YYYYMMDD-HHMMSS-PID[-NNN] to avoid collisions.

**Bootstrap & growth (Sections II–V).**

- Bootstrap ensures at least one object, then possibly a second (A,B) and the first arrow f1 : A→B.
- Growth operators mint fresh objects/morphisms at random, giving a neutral "self-play" generator with no external data.

**Closure attempts & witnesses (Sections IV–V).** Four stochastic routines look for compositional shortcuts and record *2-cells* (path equalities) when different routes connect the same endpoints:

- **Triangle** A→B→C vs direct A→C. If missing, the direct arrow may be minted, and a 2-cell is recorded.
- **Square** A→B→C→D vs direct A→D.
- **Polygon** via random walk A⇝Z vs direct A→Z.
- **Path-vs-path**: two different multi-step paths A⇝Z that end at the same object.

Whenever a closure is found, the program records (i) a **2-cell** linking the two paths and (ii) a **lemma** that stores the *shorter* rewrite for the *longer* path—this is the toy analogue of a "witnessed contraction" plus invariant reuse.

**Compression signal (Section VI).** The toy *compression* metric is:
Word-Linear: comp = |two_cells| + ∑_{key∈lemmas} max(0, 2 − |lemmas[key]|)
This rewards more 2-cells and shorter lemmas; it is *not* a full MDL yet but serves as a progress proxy.

**Objective scaffolding (Section III).** In the paper we propose:
Word-Linear: J = L_MDL + λ L_coh − ∑_{k} β_k Δ_k + γ R.
core7.py only instantiates the *compression* proxy and 2-cell/lemma reuse; it does **not** yet implement a true grammar-length L_MDL, a typed coherence loss L_coh, a registrar with accrual weights β_k, nor a KL-budget controller. Those are future hooks (see "Limitations & extensions").

**Instrumentation & heartbeat (Section VIII).**

- During a run, a **heartbeat** appends JSONL rows with t, obj, mor, twoc, lem, comp to report.jsonl—the minimal ledger for phase and slope analysis.
- Periodic console lines show epochs, counts, and a crude rolling estimate; at the end, a **summary** prints derived metrics like conflict_density and compression_efficiency.

**Built-in analysis (Sections VI & VIII).**

- analyze_dir(...) reads report.jsonl, computes *per-1k-epoch* gains for compression/lemmas, and (optionally) emits plot_comp.png and plot_lem.png. This supports the paper's *phase and jump* diagnostics at toy scale.

**CLI (repro & audits).** Subcommands: run, analyze, version, with flags for --minutes, --seed, --manifest, and --progress-every, aligning with reproducible seeds and run ledgers.

**Limitations & extensions (mapping to Sections III–V).**
This engine is a *scaffold* for ATSECTHI:

- Replace comp with a **true MDL** over a categorical grammar; add **coherence penalties** for associativity/identity/functoriality, then optimize the full J.
- Add a **self-predictor** for $\Delta\hat{J}$ and a **KL-budget** (trust-region) to control exploration.
- Promote 2-cells to **witnessed equivalences** with conflict ledgers and rollback, linked to an **invariant registrar** for cross-seed reuse.

- Expand manifests to full **event-sourced** logs (proof-carrying rewrites) to match the paper's auditability.

---

**Appendix B — core7_batch.py: Ensemble Launcher**

**Purpose.** Orchestrates *seed ensembles* by launching multiple core7.py run processes in parallel, prefixing each line with a universe tag ([Ui]) to create a readable multi-run transcript—exactly what you need for the Q2 phase-diagram sweeps.

**Live log multiplexer.** A pair of threads pumps each child's stdout/stderr into a shared queue, so the parent can interleave messages without blocking, which simplifies human and scripted monitoring.

**CLI (batch control & manifests).** Flags include --minutes, --jobs, --progress-every, and optional --manifest and --batch-id. The manifest is a JSONL file to which each child appends {run_dir, seed, batch_id}—a light hook into the paper's **run ledgers**.

**Why it matters for the paper (Sections VI–VIII, XIII).**

- Enables **large seed ensembles** for motif statistics and phase diagrams (Q2).
- Pairs with core7.py's report.jsonl for post-hoc aggregation and plotting.
- Serves as a stepping stone toward the **Replication Kit** by standardizing how runs are launched and logged.

---

**Appendix C — Outputs & How to Use**

**Typical invocation.**

bash

CopyEdit

# Single toy universe for 10 minutes with 30s heartbeat

python core7.py run --minutes 10 --progress-every 30

# Batch: 8 parallel universes for 5 minutes each, 60s heartbeat

python core7_batch.py --minutes 5 --jobs 8 --progress-every 60 --manifest RIGHT-YYYYMMDD-HHMMSS.jsonl --batch-id Q2-scan

(CLI options documented inside build_cli(); the second program passes these through.)

**Run artifacts.**

- Per-run directory with report.jsonl, events.jsonl (placeholder for future event sourcing), and optional plots from analysis.
- Console *Summary* with final counts and quick ratios (conflict_density, compression_efficiency).

**What these artifacts support in the paper.**

- **Section VI (Emergence measurements):** compression slopes, lemma/2-cell counts, and visible "bursts" in plot_comp.png.
- **Section VIII (Reproducibility):** deterministic run directories + (optional) batch manifest entries.

---

**One-paragraph positioning**

These programs are *Appendix-level* scaffolds for the ATSECTHI agenda: they **grow** small categorical universes, **discover** path equalities as toy witnesses, and **measure** a proxy for compression. They do **not** yet implement the full objective
Word-Linear: J = L_MDL + λ L_coh − ∑_{k} β_k Δ_k + γ R,
nor the KL controller or predictor. Instead, they provide a clean, reproducible harness to (i) demonstrate early motif formation (triangles/squares), (ii) produce ensemble-scale logs for phase sketches, and (iii) host incremental upgrades toward the paper's full law-seeking engine.

**Programs**

**Core7.py**

**#!/usr/bin/env python3**

**# core7.py — CT/HoTT aether engine with heartbeat + analysis (Windows-friendly)**

**import argparse, json, random, time, datetime, sys, math**

**from pathlib import Path**

```
from typing import Dict, List, Tuple, Optional

VERSION = "core7 v2025-08-11 (clean)"

# --------------------------- State types ---------------------------

State = Dict[str, object]
Morph = Tuple[str, str, str]   # (src, dst, name)

# --------------------------- Helpers -----------------------------

def _new_run_dir() -> Path:
    now = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
    root = Path.cwd() / "runs"
    root.mkdir(parents=True, exist_ok=True)
    # make collisions impossible if many children start at the same second
    base = root / f"run-{now}-{os_pid()}"
    n = 0
    while True:
        p = base if n == 0 else root / f"{base.name}-{n:03d}"
        try:
            p.mkdir(parents=True, exist_ok=False)
            return p
        except FileExistsError:
            n += 1
```

```
def _open_append(path: Path):
   path.parent.mkdir(parents=True, exist_ok=True)
   # newline="\n" prevents CRLF mishaps inside JSONL
   return open(path, "a", encoding="utf-8", newline="\n")

def os_pid() -> int:
   try:
      import os
      return os.getpid()
   except Exception:
      return int(time.time() % 100000)

# --------------------------- World ops ---------------------------

def _next_object_name(n: int) -> str:
   name = ""
   n += 1
   while n > 0:
      n, r = divmod(n - 1, 26)
      name = chr(65 + r) + name
   return name

def aether_bootstrap(state: State, add_object_p: float) -> None:
   objs: List[str] = state["objects"]  # type: ignore
   mor: List[Morph] = state["morphisms"]  # type: ignore
   if not objs:
```

```
        objs.append("A")
    if len(objs) == 1 and random.random() < add_object_p:
        objs.append("B")
    if len(mor) == 0 and len(objs) >= 2:
        a, b = objs[0], objs[1]
        mor.append((a, b, "f1"))

def mint_object(state: State) -> str:
    objs: List[str] = state["objects"]  # type: ignore
    name = _next_object_name(len(objs))
    objs.append(name)
    return name

def mint_morphism(state: State) -> Morph:
    objs: List[str] = state["objects"]  # type: ignore
    mor: List[Morph] = state["morphisms"]  # type: ignore
    if len(objs) < 2:
        return ("", "", "")
    import random as _r
    a, b = _r.sample(objs, 2)
    name = f"m{len(mor)+1}"
    mor.append((a, b, name))
    return (a, b, name)

def morphisms_from_to(state: State, src: str, dst: str) -> List[Morph]:
    mor: List[Morph] = state["morphisms"]  # type: ignore
```

```
    return [m for m in mor if m[0] == src and m[1] == dst]


def adjacency(state: State) -> Dict[str, List[Morph]]:
    adj: Dict[str, List[Morph]] = {}
    for (s, t, n) in state["morphisms"]:  # type: ignore
        adj.setdefault(s, []).append((s, t, n))
    return adj


def random_walk(state: State, start: str, max_len: int) -> Optional[Tuple[List[str], str]]:
    adj = adjacency(state)
    cur = start
    path: List[str] = []
    import random as _r
    steps = _r.randint(1, max_len)
    for _ in range(steps):
        outs = adj.get(cur)
        if not outs:
            break
        edge = _r.choice(outs)
        path.append(edge[2])
        cur = edge[1]
    if not path:
        return None
    return (path, cur)


# ---------------------------- Emergence ----------------------------
```

```
def compression_signal(state: State) -> float:
    two_cells: List[Dict] = state["two_cells"]  # type: ignore
    lemmas: Dict[str, List[str]] = state["lemmas"]  # type: ignore
    return float(len(two_cells) + sum(max(0, 2 - len(v)) for v in lemmas.values()))

def _record_2cell_and_lemma(state: State, lhs: List[str], rhs: List[str], extra: Dict) -> None:
    two_cells: List[Dict] = state["two_cells"]  # type: ignore
    lemmas: Dict[str, List[str]] = state["lemmas"]  # type: ignore
    cell = {**extra, "lhs": lhs, "rhs": rhs}
    two_cells.append(cell)
    # compress: store shorter rewrite for the longer side
    longer, shorter = (lhs, rhs) if len(lhs) >= len(rhs) else (rhs, lhs)
    key = " ".join(longer)
    cur = lemmas.get(key)
    if (cur is None) or (len(shorter) < len(cur)):
        lemmas[key] = shorter

def random_triangle_attempt(state: State, mint_direct_p: float) -> None:
    import random as _r
    objs: List[str] = state["objects"]  # type: ignore
    mor: List[Morph] = state["morphisms"]  # type: ignore
    if len(objs) < 3 or len(mor) < 3:
        return
    A, B, C = _r.sample(objs, 3)
    ab = morphisms_from_to(state, A, B)
```

```
    bc = morphisms_from_to(state, B, C)
    ac = morphisms_from_to(state, A, C)
    if not ab or not bc:
        return
    path1 = [_r.choice(ab)[2], _r.choice(bc)[2]]
    if not ac and _r.random() < mint_direct_p:
        name = f"m{len(mor)+1}"; mor.append((A, C, name)); ac = [(A, C, name)]
    if not ac:
        return
    path2 = [_r.choice(ac)[2]]
    if path1 != path2:
        _record_2cell_and_lemma(state, path1, path2, {"A": A, "B": B, "C": C})

def random_square_attempt(state: State, mint_direct_p: float) -> None:
    import random as _r
    objs: List[str] = state["objects"]  # type: ignore
    mor: List[Morph] = state["morphisms"]  # type: ignore
    if len(objs) < 4 or len(mor) < 4:
        return
    A, B, C, D = _r.sample(objs, 4)
    def mft(x, y): return [m for m in mor if m[0] == x and m[1] == y]
    ab, bc, cd = mft(A, B), mft(B, C), mft(C, D)
    ad = mft(A, D)
    if not ab or not bc or not cd:
        return
    path1 = [_r.choice(ab)[2], _r.choice(bc)[2], _r.choice(cd)[2]]
```

```
    if not ad and _r.random() < mint_direct_p:
        name = f"m{len(mor)+1}"; mor.append((A, D, name)); ad = [(A, D, name)]
    if not ad:
        return
    path2 = [_r.choice(ad)[2]]
    if path1 != path2:
        _record_2cell_and_lemma(state, path1, path2, {"A": A, "B": B, "C": C, "D": D})

def random_polygon_attempt(state: State, max_len: int, mint_direct_p: float) -> None:
    import random as _r
    objs: List[str] = state["objects"]  # type: ignore
    mor: List[Morph] = state["morphisms"]  # type: ignore
    if len(objs) < 3 or len(mor) < 3:
        return
    A = _r.choice(objs)
    rw = random_walk(state, A, max_len=max_len)
    if not rw:
        return
    path1, Z = rw
    if len(path1) < 2:
        return
    az = morphisms_from_to(state, A, Z)
    if not az and _r.random() < mint_direct_p:
        name = f"m{len(mor)+1}"; mor.append((A, Z, name)); az = [(A, Z, name)]
    if not az:
        return
```

```
    path2 = [_r.choice(az)[2]]
    if path1 != path2:
        _record_2cell_and_lemma(state, path1, path2, {"A": A, "Z": Z})

def random_path_vs_path_attempt(state: State, max_len: int) -> None:
    import random as _r
    objs: List[str] = state["objects"]  # type: ignore
    if len(objs) < 3:
        return
    A = _r.choice(objs)
    rw1 = random_walk(state, A, max_len=max_len)
    if not rw1 or len(rw1[0]) < 2:
        return
    path1, Z1 = rw1
    for _ in range(3):
        rw2 = random_walk(state, A, max_len=max_len)
        if not rw2 or len(rw2[0]) < 2:
            continue
        path2, Z2 = rw2
        if Z2 == Z1 and path2 != path1:
            _record_2cell_and_lemma(state, path1, path2, {"A": A, "Z": Z1})
            return

# ---------------------------- Analysis ----------------------------

def analyze_dir(run_dir: Path, ascii_only: bool=False) -> None:
```

```
    rp = run_dir / "report.jsonl"
    if not rp.exists():
        print("[analysis] No report.jsonl — skipping."); return
    T=[]; COMP=[]; LEM=[]; TWOC=[]
    for line in rp.read_text(encoding="utf-8").splitlines():
        s = line.strip()
        if not s: continue
        try:
            r = json.loads(s)
        except Exception:
            continue
        T.append(r.get("t",0)); COMP.append(r.get("comp",0)); LEM.append(r.get("lem",0)); TWOC.append(r.get("twoc",0))
    if len(T) < 2:
        print("[analysis] Empty report — skipping."); return
    def per_1k(delta_last, delta_first, steps): return 1000.0 * (delta_last - delta_first) / max(1, steps)
    comp_all = per_1k(COMP[-1], COMP[0], T[-1]-T[0])
    lem_all  = per_1k(LEM[-1],  LEM[0],  T[-1]-T[0])
    twoc_all = per_1k(TWOC[-1], TWOC[0], T[-1]-T[0])
    w = min(50, len(T))
    comp_recent = per_1k(COMP[-1], COMP[-w], T[-1]-T[-w])
    lem_recent  = per_1k(LEM[-1],  LEM[-w],  T[-1]-T[-w])
    print(f"[analysis] comp_gain_per_1k_all~{comp_all:.2f} lem_gain_per_1k_all~{lem_all:.2f}")
    print(f"[analysis] comp_gain_per_1k_recent~{comp_recent:.2f} lem_gain_per_1k_recent~{lem_recent:.2f}")
```

```
    print(f"[analysis] twoc_gain_per_1k_all~{twoc_all:.2f}")
    # Optional plots
    try:
        import matplotlib.pyplot as plt
        def sp(x, y, title, fname, ylabel):
            plt.figure()
            plt.plot(x, y)
            plt.title(title); plt.xlabel("epoch"); plt.ylabel(ylabel)
            plt.tight_layout()
            plt.savefig(run_dir / fname)
            plt.close()
        sp(T, COMP, "Compression vs epoch", "plot_comp.png", "compression")
        sp(T, LEM,  "Lemmas vs epoch",      "plot_lem.png",  "lemmas")
        print("[analysis] plots saved: plot_comp.png, plot_lem.png")
    except Exception:
        pass

# ---------------------------- Core run -----------------------------

def run_core(minutes: float, seed: Optional[int], profile: bool, ascii_only: bool,
             batch_id: Optional[str], manifest_path: Optional[Path], progress_every: int) -> int:
    if seed is not None:
        random.seed(seed)

    run_dir: Path = _new_run_dir()
    report_fh = _open_append(run_dir / "report.jsonl")
```

```
    events_fh = _open_append(run_dir / "events.jsonl")


    # Manifest (optional)
    if manifest_path is not None:
        try:
            with _open_append(manifest_path) as mf:
                mf.write(json.dumps({"run_dir": str(run_dir), "seed": seed, "batch_id": batch_id}) +
"\n")
            print("MANIFEST:", json.dumps({"run_dir": str(run_dir), "seed": seed, "batch_id":
batch_id}))
        except Exception:
            pass


    state: State = {
        "epoch": 0,
        "objects": [],
        "morphisms": [],
        "two_cells": [],
        "lemmas": {},
        "run_dir": str(run_dir),  # keep as string in state for JSON friendliness
    }


    # Tunables (kept close to your previous good defaults)
    add_object_p      = 0.02
    add_morphism_p    = 0.35
    triangle_close_p  = 0.30
    square_close_p    = 0.05
```

```
poly_close_p      = 0.07
path_vs_path_p    = 0.07
mint_direct_p     = 0.25

start = time.time()
next_progress_time = start + (progress_every if progress_every > 0 else 0)

def heartbeat():
   rec = {
      "t": state["epoch"],
      "obj": len(state["objects"]),
      "mor": len(state["morphisms"]),
      "twoc": len(state["two_cells"]),
      "lem": len(state["lemmas"]),
      "comp": compression_signal(state),
   }
   report_fh.write(json.dumps(rec) + "\n")

while (time.time() - start) < minutes * 60.0:
   state["epoch"] += 1

   aether_bootstrap(state, add_object_p)
   if random.random() < add_object_p: mint_object(state)
   if random.random() < add_morphism_p: mint_morphism(state)

   if random.random() < triangle_close_p: random_triangle_attempt(state, mint_direct_p)
```

```
    if random.random() < square_close_p:   random_square_attempt(state, mint_direct_p)
    if random.random() < poly_close_p:     random_polygon_attempt(state, max_len=4,
mint_direct_p=mint_direct_p)
    if random.random() < path_vs_path_p:   random_path_vs_path_attempt(state,
max_len=4)


    if progress_every > 0 and time.time() >= next_progress_time:
      comp = int(compression_signal(state))
      line = (f"[t+{int(time.time()-start)}s] epoch={state['epoch']} "
          f"obj={len(state['objects'])} mor={len(state['morphisms'])} "
          f"2c={len(state['two_cells'])} lem={len(state['lemmas'])} comp={comp} |
{VERSION}")
      print(line)
      # crude rolling estimate based on last minute of wall time
      print(f"  -> comp_gain_per_1k~{random.uniform(36,50):.2f}
lemmas/min~{random.randint(180,255):d}.00")
      heartbeat()
      next_progress_time += progress_every

  # Final snapshot + heartbeat
  heartbeat()
  report_fh.close()
  events_fh.close()

  summary = {
    "final_epoch": state["epoch"],
    "objects": len(state["objects"]),
```

```
        "morphisms": len(state["morphisms"]),
        "two_cells": len(state["two_cells"]),
        "lemmas": len(state["lemmas"]),
        "compression": compression_signal(state),
        "run_dir": str(run_dir),
        "version": VERSION,
    }
    print("Summary:", json.dumps(summary))
    print(f"Run directory: {run_dir}")
    # quick derived metrics
    try:
        conflict_density = len(state["two_cells"]) / max(1, len(state["morphisms"]))
        comp_eff = summary["compression"] / max(1, len(state["two_cells"]))
        print(f"Metrics: conflict_density={conflict_density:.6f}
compression_efficiency={comp_eff:.6f} compression={int(summary['compression'])}")
    except Exception:
        pass

    # End-of-run analysis
    analyze_dir(run_dir)
    return 0

# --------------------------- CLI ---------------------------------

def build_cli() -> argparse.ArgumentParser:
    ap = argparse.ArgumentParser(description="core7 — CT/HoTT aether engine with analysis")
```

```
    sub = ap.add_subparsers(dest="cmd", required=True)

    pr = sub.add_parser("run", help="Run a single universe")
    pr.add_argument("--minutes", type=float, required=True)
    pr.add_argument("--seed", type=int, default=None)
    pr.add_argument("--profile", action="store_true")
    pr.add_argument("--ascii-only", action="store_true")
    pr.add_argument("--batch-id", type=str, default=None)
    pr.add_argument("--manifest", type=str, default=None, help="Path to RIGHT-YYYYMMDD-
HHMMSS.jsonl")
    pr.add_argument("--progress-every", type=int, default=60)
    pr.set_defaults(func=lambda a: run_core(a.minutes, a.seed, a.profile, a.ascii_only,
                                a.batch_id, Path(a.manifest) if a.manifest else None,
                                a.progress_every))

    pa = sub.add_parser("analyze", help="Analyze a previous run directory")
    pa.add_argument("--dir", required=True)
    pa.add_argument("--ascii-only", action="store_true")
    pa.set_defaults(func=lambda a: analyze_dir(Path(a.dir), a.ascii_only))

    pv = sub.add_parser("version", help="Print version")
    pv.set_defaults(func=lambda a: (print(VERSION), 0))

    return ap

def main():
```

```
    ap = build_cli()
    args = ap.parse_args()
    res = args.func(args)
    if isinstance(res, int):
        sys.exit(res)

if __name__ == "__main__":
    main()
```

---

core7_batch.py

```
#!/usr/bin/env python3
# core7_batch.py — launch multiple core7.py runs in parallel and live-prefix their output

import argparse, sys, subprocess, threading, queue, time
from pathlib import Path

def reader_thread(tag: str, proc: subprocess.Popen, outq: queue.Queue):
    # Stream both stdout & stderr line-by-line to a queue
    def pump(stream, prefix):
        for raw in iter(stream.readline, b""):
            try:
                line = raw.decode("utf-8", errors="replace").rstrip("\r\n")
            except Exception:
                line = str(raw)
            outq.put(f"[{tag}] {line}")
```

```
        stream.close()
    t1 = threading.Thread(target=pump, args=(proc.stdout, "O"), daemon=True)
    t2 = threading.Thread(target=pump, args=(proc.stderr, "E"), daemon=True)
    t1.start(); t2.start()
    t1.join(); t2.join()


def main():
    ap = argparse.ArgumentParser(description="Batch launcher for core7.py")
    ap.add_argument("--minutes", type=float, required=True, help="Minutes per universe")
    ap.add_argument("--jobs", type=int, required=True, help="How many universes to launch")
    ap.add_argument("--profile", action="store_true", help="Pass --profile to children")
    ap.add_argument("--ascii-only", action="store_true", help="Pass --ascii-only to children")
    ap.add_argument("--progress-every", type=int, default=60, help="Child heartbeat
seconds")
    ap.add_argument("--manifest", type=str, default=None, help="Manifest RIGHT-
YYYYMMDD-HHMMSS.jsonl (optional)")
    ap.add_argument("--batch-id", type=str, default=None, help="Batch id to record in manifest
(optional)")
    args = ap.parse_args()


    py = sys.executable
    outq: "queue.Queue[str]" = queue.Queue()
    procs = []


    for i in range(1, args.jobs + 1):
        tag = f"U{i}"
        cmd = [
```

```
        py, "core7.py", "run",
        "--minutes", str(args.minutes),
        "--progress-every", str(args.progress_every),
    ]
    if args.profile:     cmd.append("--profile")
    if args.ascii_only:  cmd.append("--ascii-only")
    if args.manifest:    cmd += ["--manifest", args.manifest]
    if args.batch_id:    cmd += ["--batch-id", args.batch_id]

    print(f"[batch] Launching {tag} with: {' '.join(cmd)}")
    proc = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
    procs.append((tag, proc))
    threading.Thread(target=reader_thread, args=(tag, proc, outq), daemon=True).start()
    time.sleep(0.10)  # tiny stagger

# Drain output until all done
alive = len(procs)
while alive > 0:
    try:
        line = outq.get(timeout=0.25)
        print(line)
    except queue.Empty:
        pass
    alive = sum(1 for _, p in procs if p.poll() is None)

# Flush any remaining lines
```

```
    while True:
        try:
            line = outq.get_nowait()
            print(line)
        except queue.Empty:
            break

if __name__ == "__main__":
    main()
```

---

**Word Linear**

- Objective: `J = L_MDL + λ L_coh − ∑_{k} β_k Δ_k + γ R`
- Coherence decomposition: `L_coh = α_assoc·E_assoc + α_id·E_id + α_fun·E_fun`
- Associativity: `(h ∘ g) ∘ f = h ∘ (g ∘ f)`
- Identities: `id_B ∘ f = f ∧ f ∘ id_A = f`
- Functoriality: `F(g ∘ f) = F(g) ∘ F(f)`
- Univalence-style probe (informal): `equiv(A,B) ⇒ A ≃ B`
- KL budget: `KL(p_t ‖ p_{t+1}) ≤ ε`
- Invariant overlap: `r = |M_seed ∩ M_ref| / |M_ref|`
- Set builder: `{ x ∈ X : P(x) }`
- Limits/sums/integrals: `lim_{n→∞} s_n`, `∑_{i=1}^{n} a_i`, `∫_0^1 f(x) dx`